VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Čítání osob ve videosekvencích

Counting the Persons in Videosequences

2015 Václav Adamec

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Václav Adamec**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Čítání osob ve videosekvencích
Counting the Persons in Videosequences

Zásady pro vypracování:

Cílem diplomové práce je vytvořit software pro analýzu videosekvencí obsahujících pohybující se lidské postavy. Analýzou se rozumí zjištění počtu prošedších osob, případně pak také detekce některých neobvyklých stavů, jako jsou například pády osob (tato druhá část je ovšem nepovinná; diplomant se pro ni může rozhodnout na základě svého zájmu a časových možností). V diplomové práci proveďte následující:

1. Navrhněte/zvolte vhodnou metodu detekce postav ve videosekvencích.
2. Navrhněte způsob počítání osob (např. metodou vyhodnocování trajektorií), případně také navrhněte metodu detekce zvláštních situací.
3. Navržené řešení realizujte v C/C++.
4. Řešení řádně experimentálně prověřte.

Způsob a výsledky řešení popište v textové části práce.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Dr. Ing. Eduard Sojka**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2015

Adamec

Rád bych poděkoval vedoucímu mé diplomové práce panu doc. Dr. Ing. Eduardu Sojkovi za cenné rady při vytváření této práce. Děkuji také všem, kteří mě podporovali po celou dobu studia.

Abstrakt

Cílem práce bylo vytvoření softwaru pro počítání osob ve videosekvencích. Navržený algoritmus využívá k segmentaci postav detekci pohybu pomocí metody Mixture of Gaussians. Funkce počítání osob je založena na sledování trajektorie postav s využitím Kalmanova filtru.Pro zvýšení robustnosti aplikace v případě částečného překrytí sledovaných osob je využita metoda porovnání se vzorem. Výsledné řešení je implementováno v programovacím jazyce C++ s využitím knihovny OpenCV.

Klíčová slova: počítání osob, sledování trajektorie, Mixture of Gaussians, Kalmanův filtr

Abstract

The aim of this thesis was to implement software for counting the persons in videosequences. The proposed algorithm uses an motion detection method Mixture of Gaussians to segment the persons. People counting function is based on trajectory tracking using the Kalman filter. The template matching metod is used to increase the robust of application in the case of partial occlusion of observed individuals. The solution is implemented in C ++ programing language, using the OpenCV library.

Keywords: counting the persons, trajectory tracking, Mixture of Gaussians, Kalman filter

Seznam použitých zkratek a symbolů

MOG	–	Mixture of Gaussians
ASM	–	Active Shape Model
HOG	–	Histogram of Oriented Gradiens
MVFI	–	Motion Vector Flow Instance

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Souběžně s rozsahem dnešních kamerových systémů narůstají i nároky na obsluhu konající dohled. V dohledovém kamerovém systému čítajícím desítky kamer je technologicky a personálně velice náročné, až nemožné, pokrýt veškeré záběry. Proto je logickým krokem tuto činnost zcela automatizovat. Automatizací dochází ke snížení nákladů na provoz a také k eliminování lidského faktoru. Strojové zpracování obrazu tak přináší nepopiratelné výhody, stále však má své limity. Pro dosažení sofistikovanější funkcionality je kritická schopnost správné detekce a trasování objektů zájmu (lidské postavy) v prostoru scény a v čase. Pro lidské oko a mozek je tento problém na základě našich zkušeností často intuitivní a triviální, softwarové řešení je však stále objektem výzkumu. Dřívější systémy se většinou vyznačovaly jednoduššími funkcemi jako je např. detekce pohybu atd. U těch dnešních je snaha o hlubší porozumění sledované scény. Příkladem může být vytvoření modelu typického chování objektů ve scéně. Jakákoli odchylka od běžného chování je pak detekována.

Typickými místy pro využití těchto systémů jsou například nákupní centra, nádražní haly, prostory metra, ale i venkovní prostory, jako jsou sportovní stadiony, parkoviště a v neposlední řadě i centra měst. Systém obsluze místo náhodně vybraných kamer nabídne pohledy kamer zabírajících scénu, ve které se děje něco atypického. Obsluha má tak možnost situaci vyhodnotit a adekvátně reagovat. Systém, který dokáže odhalit nezvyklé chování, pak může sloužit k včasnému varovaní před možnou kriminalitou nebo třeba upozornit na člověka, který zkolaboval a nehýbe se. Využití lze nalézt i k automatizované tvorbě statistik, jako jsou počítání osob, jejich typický pohyb po scéně atd. Tyto statistiky pak mohou sloužit pro optimalizaci dopravy, ale například i pro marketingové účely. Analýzou pohybu zákazníků po obchodě můžeme stanovit typickou trasu průchodu obchodem, čas strávený u jednotlivého zboží, atd. Získané statistiky pak mohou sloužit jako cenný zdroj informací pro management obchodu a výsledkem je třeba efektivnější umístění reklamy pro cílového zákazníka obchodního centra. Obchodní řetězec Globus využívá kamerový systém nejen k počítání zákazníků, kteří vstoupili do prodejny, ale i k odhadu délky fronty u pokladen. V případě potřeby reaguje otevřením další pokladny. Dalším příkladem uplatnění detekce lidské postavy, nebo její části, může být interakce s PC. Na tuto skutečnost již poměrně dávno reagoval herní průmysl.

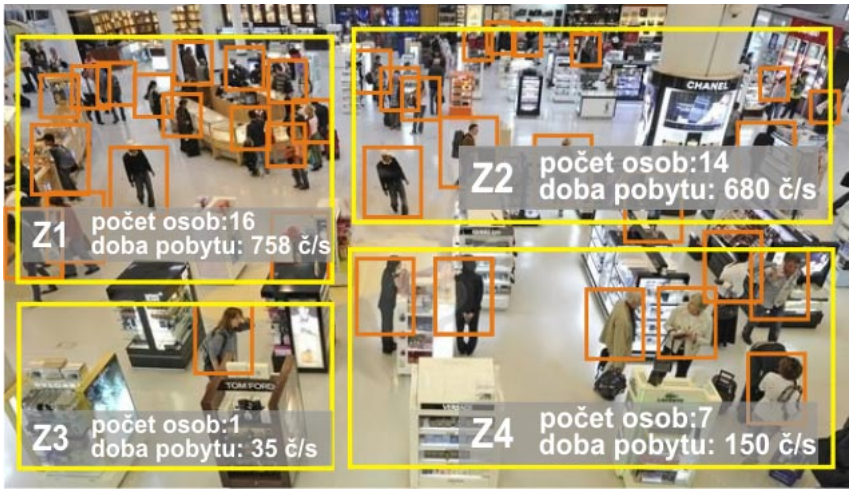
V první části této práce (kap. 2) jsou představeny metody a postupy využitelné k návrhu a implementaci základního trasovacího systému. Z těchto teoretických poznatků bylo následně čerpáno v praktické části (kap.3), kdy byla vytvořena aplikace pro sledování a počítání osob ve videosekvencích. Kap. 4 popisuje testování této aplikace a hodnotí dosažené výsledky.

2 Teoretická část

Jak již bylo nastíněno v úvodu, jsou systémy automatizované analýzy pohybu osob ve scéně v současné době velmi atraktivní. Dokladem toho je i široká nabídka mnoha firem dostupných na trhu. Za všechny můžeme jmenovat například inteligentní dohledový systém firmy NetRex [1] nebo systém Intya Counter společnosti Ronyo Technologies [2]. Produkty těchto společností reprezentují komplexní systémy určené nejen pro prodejny a obchodní centra. Disponují pokročilými funkcemi jako jsou:

- sledování osob,
- detekce pádu a nehýbající se osoby,
- počítání zákazníků na vstupu do prodejny,
- počítání zákazníků v jednotlivých zónách prodejny,
- měření doby strávené v jednotlivých zónách,
- detekce délky fronty u pokladen,
- dohled nad pokladnou.

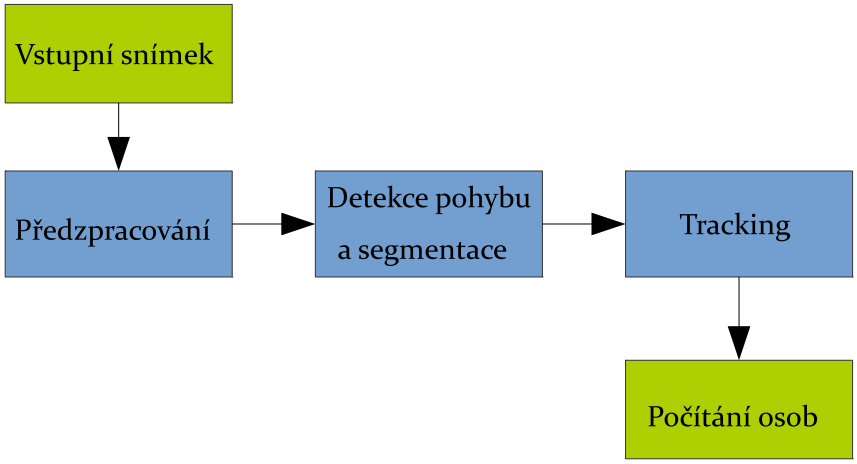
Nasazení takovéhoto systému může být poměrně silný nástroj pro snížení kriminality zákazníků, zvýšení jejich bezpečnosti a hlavně pro analýzu jejich chování v prodejně. Souhrn všech těchto aspektů pak lze využít v marketingu a poskytuje tak i určitou konkurenční výhodu.



Obrázek 1: Aplikace Intya Counter. (Převzato z [2].)

Následující kapitoly se zabývají jednotlivými kroky, na jejichž konci může být systém analyzující lidské chování z video sekvencí. Postupně jsou zde představeny vybrané metody pokrývající jednotlivé oblasti segmentace, klasifikace a kontinuálního sledování

lidské postavy v jednotlivých obrazových snímcích. Tyto kapitoly tak kopírují architekturu navrhovaného systému (obrázek 2) a vytvářejí teoretický základ pro implementaci.



Obrázek 2: Obecné schéma algoritmu počítání osob.

2.1 Detekce pohybu

V případě snímání scény s pohybem může být naší snahou rozlišit pohybující se objekty od pozadí scény. Pozadí můžeme definovat jako statickou část obrazu. Tedy pixely, které se v průběhu času nemění nebo mění svoji hodnotu jen nepatrně. Popředí (pohybující se objekty) je pak reprezentováno pixely, u kterých dochází vůči pozadí k markantní změně jejich hodnot. Na základě takto stanovených kritérií by se segmentace pohybujících objektů mohla zdát snadná. Problémy nastanou v okamžiku, kdy se pohybující objekt svými jasovými hodnotami příliš neliší od pozadí nebo je jeho pohyb příliš malý. Takový objekt , nebo jeho část, je pak nesprávně klasifikován jako součást pozadí. Mnohdy také dochází k opačnému problému, kdy jsou pixely náležící k pozadí označeny jako pohybující se. To může být způsobeno například špatnou kvalitou snímání nebo i drobnými pohyby objektů pozadí ve scéně. Typickým příkladem je pohyb listí a větví ve větru a jejich následná mylná pozitivní detekce. Primitivní metodou detekce pohybu v obraze může být vzájemné odečtení dvou snímků. Jednotlivé hodnoty pixelů snímku v čase t se odečtou od hodnot snímku v čase $t-1$ a pokud je hodnota vyšší než zvolený práh, je daný pixel označen jako pohyb. Tato metoda je velmi rychlá, ale také velmi nepřesná. Selhává v prostředí s měnícími se světelnými podmínkami a v případě, kdy se pohyb úplně zastaví. V praxi by mohla v určitých případech sloužit spíše jako detektor typu alarm a informovat, že došlo v obraze k velké změně bez hlubšího pochopení scény. Ostatní metody ke své činnosti ve většině případů využívají tzv. model prostředí. Jde o postupně vytvářený obraz pozadí scény, se kterým jsou následně jednotlivé snímky porovnávány. Podle následují vzorce můžeme rozhodnout o příslušnosti pixelu k pozadí nebo popředí.

$$|f(x,y) - b(x,y)| > T, \tag{1}$$

kde $f(x, y)$ je hodnota pixelu v aktuálním obraze, $b(x, y)$ je hodnota pixelu v modelu pozadí a T je stanovený práh. Dále v textu jsou některé metody tvorby modelu pozadí představeny blíže.

2.1.1 Průměrování

Základní technikou pro vytváření modelu prostředí je průměrování hodnot obrazových bodů na odpovídajících pozicích. Existuje mnoho variant tohoto přístupu. Model prostředí může být aktualizován s každým novým snímkem nebo po uplynutí stanovené periody. V paměti můžeme udržovat pouze jeden nebo několik snímků modelu. S tím klesá a stoupá paměťová náročnost metody. Modifikací základní metody průměrování využívající pouze jeden snímek je metoda klouzavého průměru. Příkladem metody s více snímky v paměti je metoda mediánu. Medián je určován přes všechny snímky v paměti pro každou jasovou složku zvlášť.
Výpočet klouzavého průměru:

$$B_{i+1} = \alpha \cdot F_i + (1 - \alpha) \cdot B_i, \tag{2}$$

kde B_i je předchozí snímek modelu, B_{i+1} je aktualizovaný snímek modelu, F_i je aktuální snímek a α váhový koeficient (typicky α=0,05).

2.1.2 Running Gausian Average

U této metody je pro každý obrazový bod udržována v paměti jeho střední hodnota a rozptyl. Tyto dva parametry jsou obnovovány s každým novým snímkem. O příslušnosti pixelu k pozadí se rozhoduje pomocí Gaussova (normálního) rozdělení N(μ ,σ^2) popsaného vztahem 3. Podle předpokladu nabývají pixely pozadí hodnot, které jsou popsatelné právě Gaussovým rozdělením. Pokud hodnota pixelu spadá do Gaussova rozdělení, je pixel označen jako pixel pozadí a je aktualizována střední hodnota a rozptyl modelu pozadí. Pokud se hodnota pixelu nachází nad Gaussovou křivkou, je pixel označen jako součást objektu popředí.
Gaussova funkce:

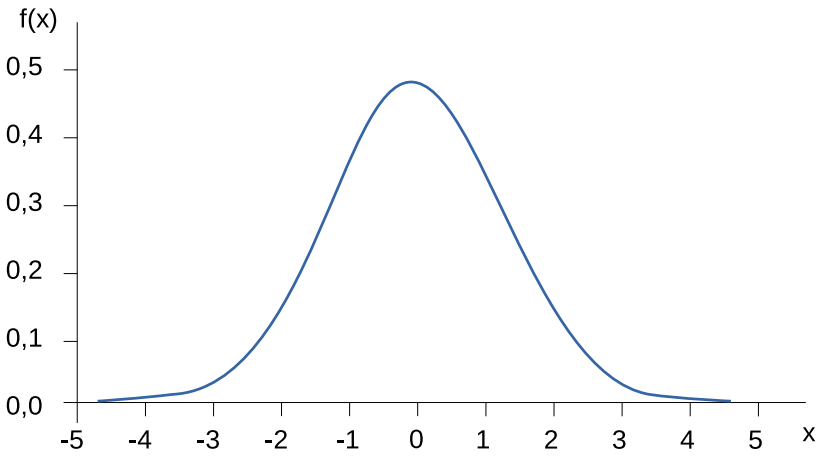
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{3}$$

V případě označení pixelu jako pozadí je přepočítána střední hodnota podle vzorce:

$$\mu_{t-1} = \alpha f_t(x, y) + (1 - \alpha)\mu_t, \tag{4}$$

kde μ je střední hodnota, $f(x, y)$ je hodnota pixelu a α je časová konstanta, pomocí které lze ovlivnit dynamiku začlenění změn scény do pozadí. Stejně tak musíme upravit i hodnotu rozptylu podle vzorce:

$$\sigma^2_{t-1} = \alpha(f_t(x, y) - \mu_t)^2 + (1 - \alpha)\sigma. \tag{5}$$



Obrázek 3: Funkce Gaussova (normálního) rozdělení - Gaussián.

2.1.3 Mixture of Gaussians

Čerpáno z [3]. Metoda Mixture of Gaussians, neboli směs gausiánů, rozvádí myšlenku modelování příslušnosti pixelu k pozadí pomocí Gaussova rozdělení ještě dále než metoda předešlá. Každý obrazový bod není v modelu reprezentován jedním, ale několika (K) Gaussovými rozděleními. Ty mají za cíl pokrýt případy, kdy pixely pozadí mohou nabývat více hodnot. Typickým případem je již zmiňovaný příklad s pohybujícími se větvemi stromu ve větru. V takovém případě pixel pozadí nabývá periodicky kromě hodnoty skutečného pozadí i hodnoty intenzity jasu překrývající větve. Mohli bychom tedy využít jeden gaussián pro popis pravděpodobnosti příslušnosti pixelu k obloze za stromem a jeden gaussián postihující pixel větve. Využitím více gaussiánů pro každý pixel tedy dokážeme eliminovat chybné označení daného pixelu jako popředí v případech opakované změny jasu způsobené např. drobným pohybem. Z uvedeného však vyplývá výpočetní a paměťová náročnost metody. Hodnoty rozložení pravděpodobnosti jsou totiž udržovány pro každý pixel a upravovány v každém snímku. Pro optimalizaci algoritmu je tedy nezbytné vhodně zvolit parametry, zejména počet gaussiánů. (V literatuře se uvádí počet 3 — 7.) V případě RGB reprezentace se každá barevná složka modeluje zvlášť. To nároky algoritmu dále zvyšuje. Za zvážení proto stojí převod obrazu na „černobílý“ do úrovní šedi. Dojde tím k urychlení algoritmu, ale za cenu snížení citlivosti v některých situacích.

Pravděpodobnost intenzity jasu pixelu X_t v čase t reprezentovaného K gaussiány lze vyjádřit vztahem:

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} \cdot \eta(X_t, \mu_{i,t}, C_{i,t}), \tag{6}$$

kde $\omega_{i,t}$ je odhad váhy i-tého gaussiánu , η je Gaussova funkce hustoty pravděpodobnosti pro n-rozměrný signál, $\mu_{i,t}$ je střední hodnota, $C_{i,t}$ je kovarianční matice i-tého gaussiánu

(v čase t).

$$\eta(X_t, \mu_{i,t}, C_{i,t}) = \frac{1}{(2\pi)^{\frac{n}{2}}|C|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu_t)^T C^{-1}(X_t-\mu_t)}. \tag{7}$$

Při aktualizaci parametrů v každém snímku je nejprve daný pixel konfrontován s gaussiány uvažovaného pixelu. V případě kdy hodnota pixelu spadá do příslušného rozdělení pravděpodobnosti (hodnota pixelu je dostatečně blízká střední hodnotě Gaussova rozdělení), je pixel přiřazen k uvažovanému rozdělení pravděpodobnosti. Dané rozdělení je následně aktualizováno pomocí předpisů:

$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t, \tag{8}$$

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t-\mu_t)^T(X_t-\mu_t), \tag{9}$$

kde

$$\rho = \alpha\eta(X_t|\mu_k, \sigma_k). \tag{10}$$

Poté je potřeba upravit váhové koeficienty jednotlivých gaussiánů aktuálního pixelu pomocí vztahu:

$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha(M_{k,t}), \tag{11}$$

kde α je konstanta vyjadřující rychlost učení modelu a $M_{k,t}$ nabvá hodnoty 1 pro přiřazený gaussián a 0 pro všechny ostatní.

Pokud není hodnota zkoumaného pixelu přiřazena do žádného rozdělení pravděpodobnosti, je rozdělení s nejmenším váhovým koeficientem smazáno a je vytvořen nový gaussián se střední hodnotou rovné hodnotě pixelu, velkým rozptylem a nízkým váhovým koeficientem. Z uvedeného vyplývá, že ne každý gaussián uvažovaného pixelu náleží popisu pozadí. Tímto způsobem se i hodnoty pixelu spadající do popředí promítnou do modelu bez jeho narušení. V případě častějšího výskytu stejných hodnot roste váhový koeficient příslušného gaussiánu. Narůstá jeho význam v modelu a objekt z popředí se tak může postupně začlenit do pozadí. Pixel jehož hodnota nespadá do žádného gaussiánu můžeme rovnou klasifikovat jako popředí. Abychom mohli rozhodnout i o pixelech s hodnotami přiřazenými k některým ze svých gaussiánů, musíme určit, které gaussiány reprezentují pozadí. Jednoduchou úvahou můžeme dojít k předpokladu, že pixely pozadí se příliš nemění nebo se mění jen pomalu. Kdežto u popředí jsou změny hodnot výraznější. Tomu budou odpovídat i příslušná rozdělení pravděpodobnosti. U gaussiánu náležícímu k pozadí můžeme očekávat vysokou váhu a malý rozptyl. Můžeme tedy sestupně seřadit gaussiány podle podílu ω/σ a následně určit kolik prvních B gaussiánů odpovídá modelu pozadí.

$$B = \underset{b}{\operatorname{argmin}} \left(\sum_{k=1}^{b} \omega_k > T \right). \tag{12}$$

2.2 Detekce lidské postavy

Rozpoznání lidské postavy ve statickém i dynamickém obraze je velice komplexní úkol. Důvodem je velká variabilita velikosti postavy, oblečení i světelných podmínek scény.

Metody detekce lidské postavy jsou proto velmi různorodé a stále jsou předmětem výzkumu. Rozdíly lze nalézt v přístupu k problému, jejich úspěšnosti i výpočetní náročnosti. Vždy však velmi záleží na typu scény a vlastnostech snímacího zařízení. Algoritmus, úspěšný v jednom případě, může v jiném naprosto selhávat. Již pouhá změna polohy kamery zapříčiní neúspěch. Je tak nutné přistupovat ke každému případu individuálně. Často však lze analýzou snímané scény a definicí omezení, které ze scény vyplývají, nalézt vhodné řešení. Takovýmto případem může být obraz ze statické kamery, kde jsou všechny osoby snímány čelně. Tady se nabízí využití detekce obličeje jako typického rysu lidské postavy. Dalšími vhodnými příznaky mohou být například detekce barvy lidské kůže a vzájemné proporční poměry částí lidského těla. Pro zvýšení robustnosti lze jednotlivé metody a přístupy kombinovat. V ideálním případě docílíme deskriptoru invariantního vůči měřítku i poloze lidské postavy v obraze.

2.2.1 Porovnání se vzorem (Template Matching)

Jde o jednoduchou metodu porovnávání prohledávaného obrazu se vzorem. Výstupem je míra podobnosti na dané pozici při posouvání šablony T po vstupním obraze I. Přes svoji jednoduchost se tato metoda k detekci lidské postavy příliš nehodí. Důvodem může být výpočetní náročnost, problematické získávání (neexistence) univerzálního vzoru, a také velká citlivost vůči rozdílné velikosti nebo natočení vzoru a prohledávaného obrazu. Částečným řešením by bylo průměrování postupně získaných vzorů a vícenásobné porovnání obrazu se vzorem s různým natočením a velikostí. Cenou za lepší detekci by byla vysoká výpočetní náročnost. Přes svoje nevýhody je zde tato metoda uvedena z důvodu jejího využití v následné implementaci projektu.

Míru podobnosti můžeme určit mnoha způsoby. Za základní lze považovat sumu absolutních rozdílů hodnot jednotlivých obrazových bodů vzoru a prohledávaného obrazu - SAD (Z anglického Sum of the Absolute Difrence).

$$R_{SAD}(x,y) = \sum_{x'} \sum_{y'} |f(x+x', y+y') - g(x', y')|. \tag{13}$$

Kde f je prohledávaný obraz, g je hledaný vzor, x,y jsou souřadnice současné pozice v obraze a x',y' jsou souřadnice vzoru. Nejlepší shodu obrazu a vzoru nalezneme podle tohoto vztahu na pozici x,y s nejmenší vypočítanou hodnotou. Další možnost porovnání šablony s obrazem popisuje následující vzorec:

$$R_{cor}(x,y) = \sum_{x} \sum_{y} (f_{(x,y)} - \bar{f})(g_{(x,y)} - \bar{g}). \tag{14}$$

Kde f je prohledávaný obraz, g je hledaný vzor, x, y jsou souřadnice současné pozice v obraze, $\bar{f}$ je střední hodnota v obraze f a $\bar{g}$ je střední hodnota jasu v obraze g. Korelační koeficient nabývá hodnot $\langle -1, 1 \rangle$ s největší shodou v 1. Rozdílné světelné podmínky ve vzoru a obraze lze redukovat použitím normalizované verze předešlého vztahu.

$$R_{cor}(x,y) = \frac{\sum_x \sum_y (f_{(x,y)} - \bar{f})(g_{(x,y)} - \bar{g})}{\sqrt{\sum_x \sum_y (f_{(x,y)} - \bar{f})^2 \sum_x \sum_y (g_{(x,y)} - \bar{g})^2}}. \tag{15}$$

2.2.2 Active Shape Model (ASM)

Je metoda navržena T. Cootsem a C. Taylorem [4], která je hojně využívána především v počítačovém zpracování biomedicínských obrazových dat. Pracuje se statistickým normalizovaným modelem (klasifikační třída vytvořená z trénovací množiny). V případě dvourozměrných dat je model určen pomocí vektoru souřadnic vhodně zvolených významných bodů. Tyto body (landmarks) jsou většinou určovány ručně v trénovací množině. Poloha a počet bodů závisí na typu (tvaru) objektu. Vzájemná poloha těchto bodů musí být invariantní vůči rotaci , translaci a změně měřítka. (Žádná z těchto transformací nesmí způsobit změnu tvaru.) Vytvořené modely normalizujeme a získáme průměrný model tvaru pomocí funkce transformace podobnosti (similarity transformation), která s modelem tvaru provádí transformace rotace o úhel θ, změnu měřítka o parametr s a posun o souřadnice x_p, y_p.

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_p \\ y_p \end{pmatrix} + \begin{pmatrix} s \cdot \cos\theta & s \cdot \sin\theta \\ -s \cdot \sin\theta & s \cdot \cos\theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}. \tag{16}$$

Protože jednotlivé normalizované modely a průměrný model tvaru nejsou stejné, je nutné vytvořit statistický model tvaru:

$$x' \approx \bar{x}' + \Phi b, \tag{17}$$

kde x je průměrný model tvaru, Φ matice vlastních vektorů získané z kovarianční matice všech trénovacích modelů a b vektor parametrů deformačního modelu. Změnou členů vektoru b lze s průměrným statistickým modelem tvaru rotovat, posouvat, měnit jeho velikost a získat tak jednotlivé modely tvaru z trénovací množiny. Pro aplikaci ASM je ještě nutné určit, jak v obraze nalézt body modelu tvaru. Proto je potřeba pro jednotlivé body definovat tzv. deskriptory. S jejich pomocí jsou pak v obraze nalezeny (upřesňovány) polohy významných bodu tvořících model. Deskriptory jsou tvořeny sadou pokud možno jednoduchých příznaků. Ty jsou většinou reprezentovány vhodnými jasovými vlastnostmi popisujícími charakteristické intenzity jasu z určitého okolí daného bodu v trénovací množině.

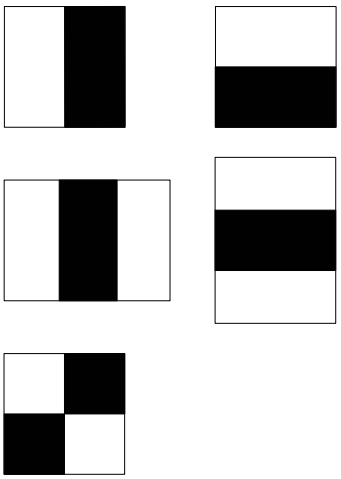
Následující sekvence popisuje konečný postup hledání daného tvaru v obraze.

1. Počáteční přibližné umístění modelu v obraze.
2. Vyhledání polohy bodů modelu v jejich blízkém okolí pomocí deskriptoru.
3. Transformace modelu podle nově nalezených poloh.
4. Opakování kroku 2 a 3 dokud dochází k úpravě modelu.

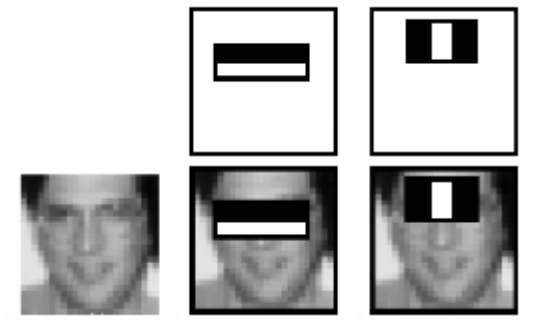
Hlavní nevýhodou metody je potřeba co nejpřesnějšího prvního umístění modelu v prohledávaném obraze. Další je pak skutečnost, že se složitostí tvaru a tedy s vyššími počty významných bodů, rostou výpočetní nároky metody.

2.2.3 Viola & Jones a AdaBoost

Detektor Viola & Jones byl poprvé představen pány Paul Viola a Michael Jones v roce 2001 v jejich práci „Robust Real-time Object detection“ [5] zabývající se detekcí obličeje. Pro názornost je na příkladu detekce obličeje představena i zde. Jejich přístup však lze samozřejmě využít i pro detekci jiných objektů, a tedy i pro detekci lidské postavy. Mezi nesporné výhody metody patří rychlost a slušná robustnost. Samotná metoda je postavena na využití jednoduchých obdélníkových filtrů, integrálním obraze a algoritmu AdaBoost.



Obrázek 4: Příznaky podobné Haarově vlnce (Haar-like features).



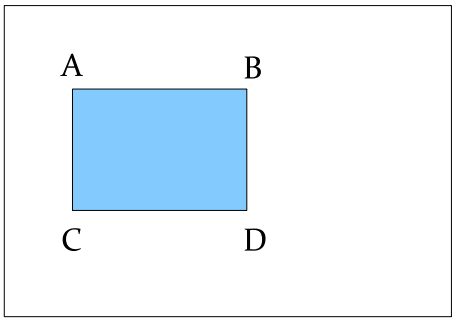
Obrázek 5: Princip použití příznaků. (Převzato z [5]).

Autoři metody vycházeli ze společných rysů lidské tváře, jako jsou tmavší oblasti očí než oblast nosu atd. Na této podstatě jsou založeny získávané příznaky, které vypočítáme odečtením součtu hodnot pixelů pod tmavým obdélníkem od součtu hodnot pixelů pod světlým obdélníkem. Tyto příznaky jsou vypočítávány pro celý obraz a v různých měřítkách. Abychom nemuseli vypočítávat součty pixelů pro každý příznak, je obraz pře-

veden na tzv. integrální obraz. Ten v každém svém bodě obsahuje součet hodnot všech předešlých pixelů.

$$I_{\sum}(x, y) = \sum_{i=1}^{x} \sum_{j=1}^{y} I(i, j). \tag{18}$$

V jednom průchodu je vypočítán integrální obraz, a pak již stačí pro výpočet sumy jakékoliv oblasti pouze čtyři přístupy do paměti a tři jednoduché výpočty. Např. pro modrou oblast na obr. 6 stačí provést $\Sigma = D - B - C + A$. Tím značně urychlíme výpočet.



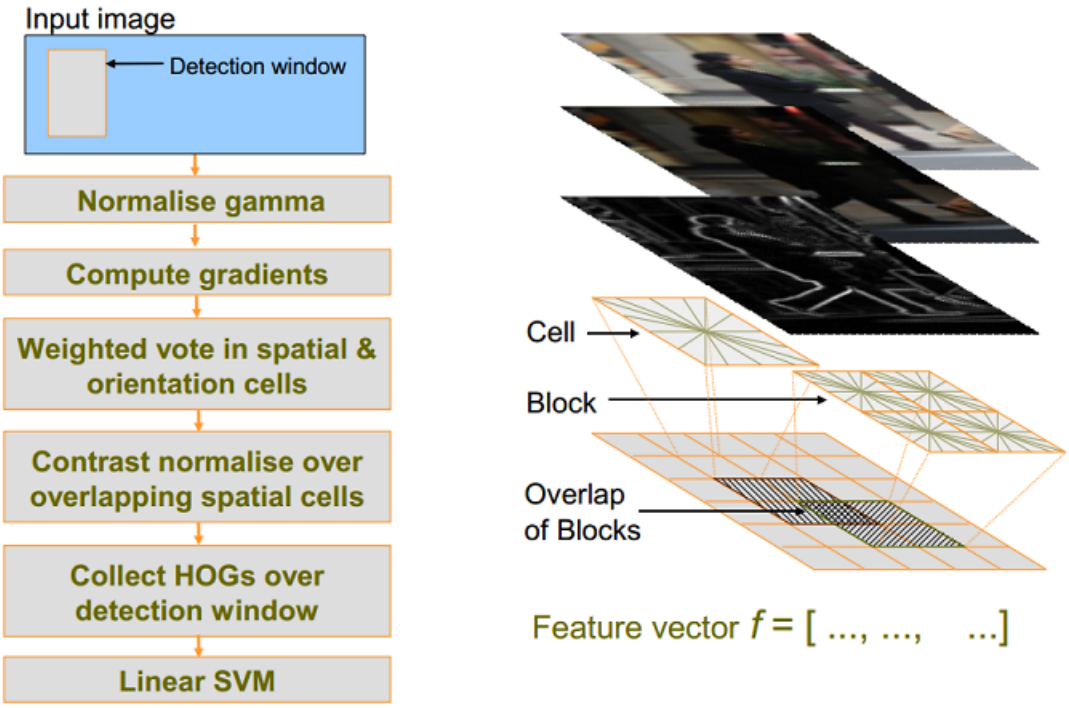
Obrázek 6: Výpočet pomocí integrálního obrazu.

Pomocí strojového učení Adaboost jsou vybrány klasifikátory sloužící pro detekci. Výsledný klasifikátor je složen z řady tzv. slabých klasifikátorů, které samostatně mají úspěšnost těsně nad hranici náhody (nad 50 %). Dohromady však tvoří jeden silný klasifikátor.

2.2.4 Histogram of Oriented Gradients (HOG)

Čerpáno z [8]. Detektor Histogram of Oriented Gradients je od počátku navržen pro detekci lidské postavy v obraze, lze jej však úspěšně využít i pro detekci jiných objektů. Metoda prokazuje lepší výsledky než předešlé metody, avšak za cenu vyšší výpočetní náročnosti. Tato metoda je založena na myšlence, že lze hledaný objekt popsat pomocí lokálních gradientů. Při implementaci algoritmu je detekční okno rozděleno na malé oblasti (buňky). Je spočítán gradient pro každý pixel v buňce a následně z těchto hodnot sestaven 1-D histogram. Z důvodu zvýšení robustnosti proti změnám osvětlení, vlivu stínů, atd., je provedena normalizace. Té je dosaženo tak, že je do buňky promítnuta informace získaná z jejího okolí, tedy ze sousedních buněk dohromady označovaných jako blok. Následuje sestavení příznakového vektoru sloužícího jako vstup pro lineární SVM.

Před samotným výpočtem gradietního obrazu můžeme provést předzpracování obrazu, například vyrovnat rozložení jasu v obraze. Za určitých okolností tak můžeme zvýšit úspěšnost metody. Podle autorů má tento krok však jen malý vliv na výsledek detekce. Oproti tomu zásadní je výpočet gradientního obrazu. Metoda předpokládá, že každý objekt je charakterizován svým tvarem, který může být v obraze reprezentován hranami.



Obrázek 7: Metoda Histogram of Oriented Gradient (Převzato z [8]).

Nalezení významných hran je tedy pro výkon detektoru stěžejní. Výsledky detektoru lze značně ovlivnit vhodnou volbou způsobu výpočtu významných lokálních hran. Obecně lze pro tuto funkci využít libovolného gradientního operátoru jakými jsou např. operátor Prewittové, Sobel atd. Při testování však nejlepší výsledky poskytovala jednoduchá derivační maska ve tvaru [-1,0,1] použitá v horizontálním i vertikálním směru. U barevného obrazu je výpočet gradientu prováděn pro každý kanál zvlášť a jako výsledná hodnota gradientu je zvolena ta nejvýraznější z jednotlivých kanálů. Jak bylo uvedeno výše, je obraz rozdělen na buňky (typicky 8×8 pixelů) a pro každou takovou buňku je stanoven histogram gradientů. To spočívá v přiřazení gradientů jednotlivých pixelů v buňce do tzv. kanálů, které rozdělují buňku v rozsahu $0\,^{\circ}$– $180\,^{\circ}$, nebo $0\,^{\circ}$– $360\,^{\circ}$. Autoři zvolili počet kanálů devět, může jich však být libovolně. Každý gradient v buňce je tedy podle své orientace přidělen do příslušného kanálu. Výsledná hodnota každého kanálu je pak rovna součtu jeho gradientů. V různých částech obrazu se mohou světelné podmínky měnit a ovlivňují tak velikosti gradientů. V dalším kroku jsou proto získané histogramy normalizovány. Normalizace probíhá nad většími bloky, které jsou tvořeny sousedními buňkami. Bloky se vzájemně překrývají a hodnoty jedné buňky se tak promítnou i do sousedních. Vektor normalizovaných histogramů z jednoho bloku představuje deskriptor. Finální klasifikace je realizována lineárním klasifikátorem vzcházejícího z SVM – Support Vector Machine. Jde tedy o příznakový klasifikátor rozdělující prostor příznaků do jednotlivých tříd pomocí nadrovin. V průběhu učení s učitelem jsou hledány podpůrné

vektory určující parametry nadrovin. Odtud název Support Vector Machine.

2.3 Sledování pohybu objektů

Pro stanovení trajektorie pohybu je nezbytné nalezení správné korespondence detekovaných objektů mezi jednotlivými snímky. Pokud se ve scéně pohybuje pouze jeden objekt, je to poměrně snadné. Při větším počtu objektů, které se pohybují různými směry, a dochází k jejich překrývání, už tento úkol tak triviální není. Dalšími negativními faktory mohou být i nepřesnost měření polohy vlivem nedokonalé segmentace, změna velikosti a vzhledu vlivem pohybu ve scéně. Za předpokladu dostatečně krátkého časového intervalu mezi jednotlivými snímky I_t a I_{t+1} bude pohyb objektu také malý a my tak můžeme omezit nové kandidátní pozice. Stačí tedy změřit vzdálenost pozice ve snímku I_t a možných nových pozic v I_{t+1}. Pokud vzdálenost splňuje stanovenou prahovou hodnotu, máme novou polohu objektu. V případě více kandidátních pozic rozhodujeme pomocí dalších charakteristik. Těmi mohou být například průměrné hodnoty jasu v určitém okolí kandidátních bodů. Za předpokladu neměnnosti vzhledu. V případě, kdy se objekt v novém snímku nepodařilo nalézt, například z důvodu zakrytí jiným objektem při křížení trajektorií, můžeme vycházet z předpokladu zachování směru a rychlosti pohybu sledovaného objektu. Na základě znalosti předešlého pohybu pak predikujeme novou pozici ve scéně. Takto můžeme dočasně nahradit skutečnou pozici do doby opětovného výskytu. K tomu můžeme využít predikční schopnost Kalmanova filtru.

2.3.1 Kalmanův filtr

Čerpáno z [6]. Kalmanův filtr je algoritmus pojmenovaný podle svého spoluautora Rudolfa Kálmána, který je využíván pro filtraci zašumělých signálů. Jeho aplikace můžeme nalézt v širokém spektru oborů jako je navigace a třeba i ekonomie. Jde o rekurzivní adaptivní filtr využívaný k modelování diskrétních dynamických systémů. Zjednodušeně lze Kalmanův filtr popsat jako predikční algoritmus, který se snaží na základě znalosti chování systému a naměřených hodnot odhadnout příští stav. Predikované hodnoty jsou pak v dalším kroku porovnány se skutečně naměřenými hodnotami. Rozdíl mezi předpokládanými a skutečně naměřenými hodnotami je následně využit k opětovnému nastavení parametrů filtru. Tato aktualizace filtru má za cíl zlepšení přesnosti nového odhadu.

Základní částí Kalmanova Filtru je stavový model dynamického systému. Tento lineární model popisuje chování sledovaného systému v čase.

$$x_{t+1} = \Phi_{t+1|t} x_t + D_t d_t + \Gamma_t w_t. \tag{19}$$

Z uvedeného vzorce vyplývá, že stav systému v čase $t+1$ je závislý na předešlém stavu x_t, budicím signálu d_t a na šumu w_t, který se společně se svým koeficientem Γ vyjadřuje nedůvěru modelu. Matice Φ, D jsou koeficienty lineární kombinace.

Další součástí modelu je model měření popisující závislost aktuálního výstupu na aktuálním stavu systému.

$$z_t = H_t x_t + v_t. \tag{20}$$

Kde matice H_t je maticí měření a člen v_t vnáší zkreslení měření vlivem šumu (chyby měření).

Chybu odhadu $\tilde{x}_t$ stanovíme jako rozdíl

$$\tilde{x}_t = x_t - \hat{x}t. \tag{21}$$

Výpočet odhadu stavu $\hat{x}_t$, tedy odhad stavového vektoru x_t může probíhat v následujících krocích.

- Predikce disperzní matice chyby odhadu

$$V_{\tilde{x},t+1|t} = \Phi_{t+1|t} V_{\tilde{x},t} \Phi_{t+1|t}^T + \Gamma_t V_{w,t} \Gamma_t^T. \tag{22}$$

- Výpočet váhové matice Kalmanova zisku

$$K_{t+1} = V_{\tilde{x},t+1|t} H_{t+1}^T [H_{t+1} V_{\tilde{x},t+1|t} H_{t+1}^T + V_{v,t+1}]^{-1}. \tag{23}$$

- Výpočet odhadu

$$\hat{x}_{t+1} = \Phi_{t+1|t} \hat{x}_t K_{t+1} z_{t+1} - H_{t+1} \Phi_{t+1|t} \hat{x}_t. \tag{24}$$

- Úprava disperzní matice

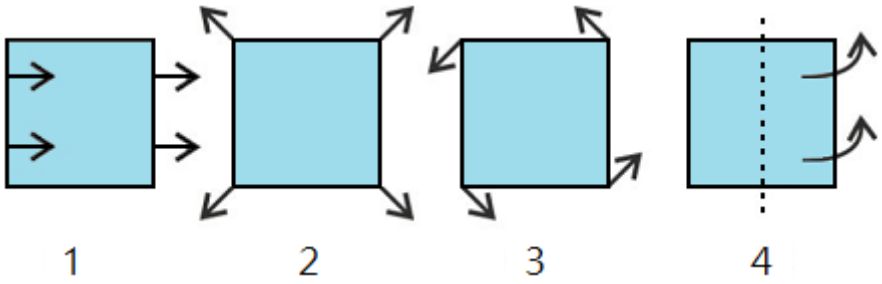
$$V_{\tilde{x},t+1} = (I - K_{t+1} H_{t+1}) V_{\tilde{x},t+1|t}. \tag{25}$$

2.3.2 Optický tok

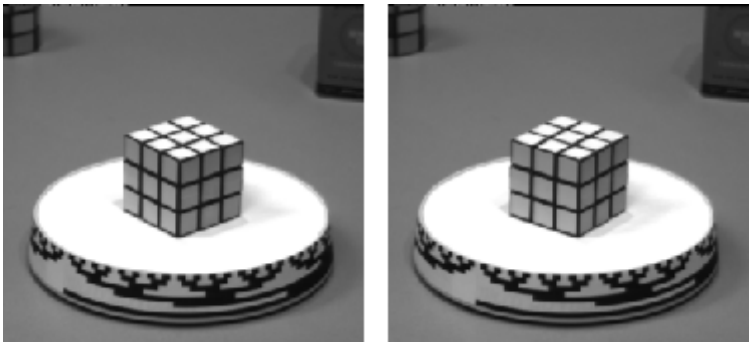
Čerpáno z [6]. Metoda optického toku zachycuje pomocí vektorového pole změny polohy jednotlivých pixelů v obraze v čase dt. Pro každý sledovaný bod je vytvořen dvourozměrný vektor rychlosti popisující jeho pohyb ve scéně mezi jednotlivými snímky. Z vytvořeného rychlostního pole můžeme posléze určit historii pohybu sledovaného objektu a případně predikovat polohu objektu v následujícím snímku. Tuto metodu lze použít nejen při scénáři, kdy se pohybuje objekt ve scéně a pozice kamery je statická, ale i při pohybu kamery. Nevýhodou metody je velká výpočetní náročnost. Ve většině případů se proto nepočítá optický tok pro všechny obrazové body, ale jen pro body nějakým způsobem pro své okolí významné (hrany atd.).

Tímto způsobem dokážeme rozlišit čtyři základní pohyby (obr.8) :

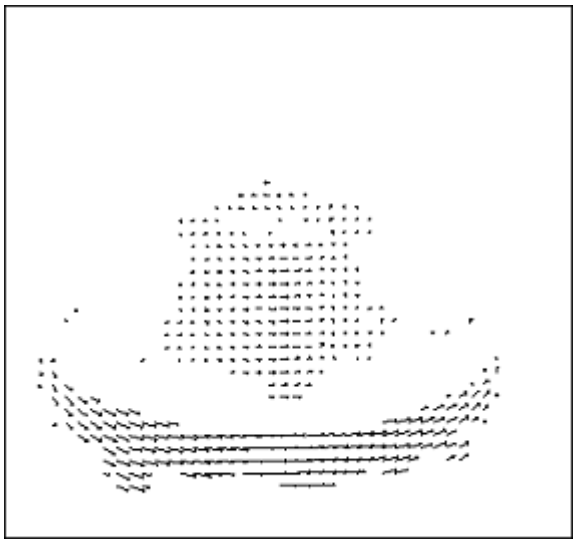
1. translační pohyb, kdy se objekt pohybuje v konstantní vzdálenosti od kamery,
2. translační pohyb směrem od nebo ke kameře,
3. rotační pohyb v konstantní vzdálenosti,
4. rotační pohyb kolmý ke kameře.



Obrázek 8: Druhy pohybu.



Obrázek 9: Rubikova kostka na otočné podložce. (Převzato z [7].)



Obrázek 10: Vektory optického toku vypočítané z obr. 9. (Převzato z [7].)

Pro správnou funkci je nutné dodržet několik předpokladů. Prvním je stejná hodnota jasů korespondenčních bodů mezi jednotlivými snímky. Dále se předpokládá jen malý

pohyb jednotlivých sledovaných bodů. δt by tedy mělo být dostatečně malé. Posledním významným předpokladem je soudržnost bodů mezi snímky. Objekt by si měl udržovat svoji strukturu.

Výpočet optického toku:
Při rozvinutí dynamické obrazové funkce v Taylorovu řadu při zanedbání vyšších řádů můžeme psát

$$f(x+\delta x, y+\delta y, t+\delta t) = f(x,y,t) + \frac{\delta f}{\delta x}\delta x + \frac{\delta f}{\delta y}\delta y + \frac{\delta f}{\delta t}\delta t. \tag{26}$$

Dále předpokládáme

$$f(x+\delta x, y+\delta y, t+\delta t) = f(x,y,t). \tag{27}$$

Po vzájemném dosazení rovnic a jejich úpravě dostaneme rovnici optického toku ve tvaru

$$f_x x' + f_y y' + f_t = 0. \tag{28}$$

2.4 Počítání osob

Následující podkapitola se nevěnuje pouze počítání osob ve videosekvencích, ale ve zkratce představí některé technické možnosti počítání osob používané v reálných instalacích. Při uplatnění v praxi dochází ve většině případů ke kombinaci více způsobů a metod. Cílem je dosáhnout co nejvyšší přesnosti a spolehlivosti. Bezpečnostní kamerový systém tak může být doplněn o různá technická zařízení. Souhrn těchto prostředků pak společně tvoří ucelený systém, který plní požadovanou funkcionalitu. Současné systémy, které se snaží vycházet pouze z analýzy obrazu, však již dokáží některé dříve používané metody úspěšně nahrazovat.

Jednou ze základních metod počítání osob je využití optické závory. Její princip spočívá v přerušení infračerveného paprsku mezi zdrojem paprsku a přijímačem. Osoba, která vstoupí do sledovaného prostoru, působí jako překážka a přeruší paprsek. To je senzorem detekováno a logický obvod zaznamená vstup. Výhodou tohoto provedení je jeho jednoduchost. Zřejmou nevýhodou je zase jeho nepřesnost vycházející z použitého principu počítání. V ideálním případě, kdy osoby procházejí jednotlivě, je detektor přesný. Pokud vstoupí do prostoru paprsku dvě osoby zároveň, zaznamená detektor pouze jeden vstup. Při přerušení paprsku první překážkou již není detektor schopen ostatní objekty zachytit. Stejný problém by nastal i při použití kamery z bočního pohledu. Osoby procházející v zákrytu nebo ve skupině nedokážeme rozlišit. Logickým řešením je zvolit vhodnější pozici snímací kamery a omezit tak splynutí více osob v jeden objekt. Vhodným umístěním na strop sledovaného prostoru, kolmo k ose směru průchodu, lze tento problém obejít. Počítané osoby procházejí přímo pod kamerou a vůbec, nebo jen minimálně, se překrývají (obrázek 11). V ideálním případě umožňuje tento postup přesnou segmentaci procházejících osob. Následně již stačí izolované objekty spočítat.

Současné inteligentní kamerové systémy využívají IP kamery s integrovanou softwarovou i hardwarovou podporou zpracování obrazu. Již samotná kamera dokáže klasifikovat různé objekty a odpadá tedy následné zpracování záznamu na serveru nebo připojeném PC. Snižují se tím nároky na přenos dat sítí i na výkon serveru. Také se nabízí

Obrázek 11: Ilustrace počítání osob. (Převzato z [2].)

využití tzv. IR kamer. Ty zachytávají infračervené záření, které člověk vyzařuje. Výsledkem je vysoce efektivní a spolehlivý systém pro počítání osob s přesností 95 – 99 % [1]. Přesnost a spolehlivost takového systému se ale také promítá do jeho pořizovací ceny.

Zde navržený způsob počítání osob se bude muset potýkat s problémy zmíněnými výše. Důvodem je pozice kamery v použitých videozáznamech. Tak jako ve všech podobných typech reálných scén i zde dochází překrytí a vzájemnému míjení procházejících osob. Uživatel aplikace bude moci pomocí konfiguračního souboru stanovit pomyslnou linii ve scéně. Detekované osoby budou po překročení stanovené linie algoritmem započítány. Přesnost počítání bude záviset zejména na přesnosti detekce postavy a následném sledování její trajektorie po scéně.

Obrázek 12: Počítací kamera.(Převzato z [1].)

2.5 Detekce pádu

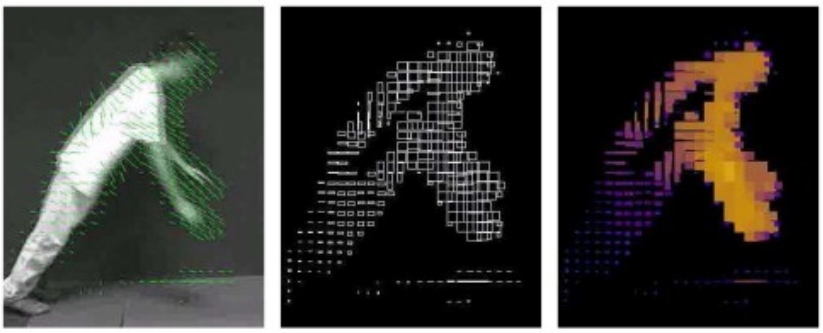
Tato část zadání není ve výsledné aplikaci implementována, přesto jsem považoval za vhodné ji zde alespoň představit. V kapitole 3.5 je i navržen algoritmus detekce pádu, který využívá zde vytvořený systém sledování osob a jejich trajektorie, a mohl by tak být dalším krokem rozšíření funkcionality aplikace.

Existuje celá řada situací a šablon chování, které můžeme chtít v dohledových systémech odhalit a adekvátně na ně reagovat. Detekce pádu rozhodně patří mezi nejvýznamnější. Téměř vždy v takové situaci jde o zdraví sledované osoby. Zejména u star-

ších osob existuje riziko pádu způsobeného například náhlou nevolností nebo problémy s pohybem. Systém schopný detekovat pád tak nalezne uplatnění nejen ve veřejných prostorech, ale hlavně v domovech pro seniory atd. Dalším cílovým prostředím mohou být například průmyslové výrobní prostory s nebezpečím úrazu. V některých případech mohou rizikové osoby využít různé technické prostředky. Jde o nositelné senzory jakými jsou akcelerometry, gyroskopy a případně mikrospínače. Senzory jsou schopné detekovat neobvyklé změny v poloze nebo rychlosti těla a mohou na to reagovat. Okolí je upozorněno např. zvukovým alarmem, nebo je odeslána informační SMS zpráva. Senzory mohou mít podobu náramku, být součásti oděvu nebo lze využít schopností dnešních chytrých telefonů. Riziková osoba však musí mít potřebný senzor stále u sebe. Obecnějším řešením je sledování celého prostoru kamerovým systémem. V případě, kdy je celý proces detekce pádu z kamer automatizován, není ani narušeno soukromí sledovaných osob. Systém sám vyhodnotí nastalou situaci a definovaným způsobem na ni reaguje.

Nejčastějším přístupem detekce pádu je analýza siluety lidské postavy. Tu získáme buď detekcí pohybu a následnou extrakcí objektů popředí nebo třeba pomocí některého z výše uvedených klasifikátorů lidské postavy. Autoři [10] využívají pro získání siluety model pozadí. O pádu je rozhodnuto na základě vlastností siluety. Zejména se sleduje úhel natočení, velikost a tvar. Tento přístup je velmi jednoduchý, ale při běžných situacích, jako je třeba sednutí osoby na židli, se může dostat do problémů.

Jinou metodu použili autoři [11]. Ve své metodě pracují s vektory rychlosti pádu. Vycházejí z předpokladu, že pád je obvykle rychlejší než běžný pohyb. Výpočtem optického toku pohybu osob vytvářejí tzv. Motion Vector Flow Instance (MVFI) vzory (obr. 13). Ty jsou poté využity k rozpoznání pádu od běžných aktivit. Zejména pohyb horní poloviny těla je při pádu výrazně rychlejší než obvyklý pohyb siluety. Některé metody toho využívají a sledují pouze pohyb hlavy. Pokud se změní rychlost a směr (zejména směr dolů), je detekován pád.



Obrázek 13: Optický tok a MVFI.(Převzato z [11].)

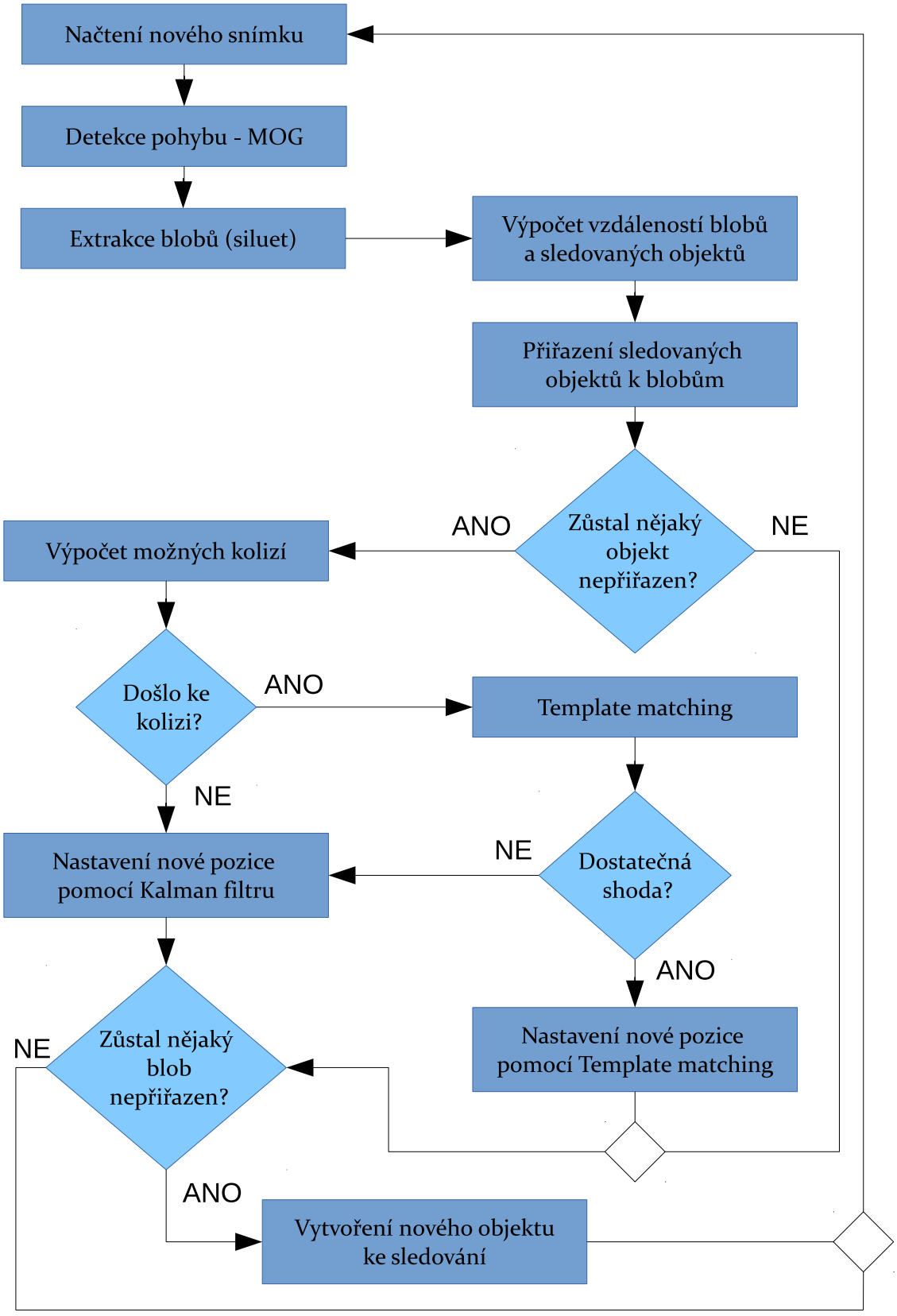
3 Praktická část

V této části práce je podrobně rozebrána vytvořená aplikace. Cílem bylo vytvořit aplikaci, která by byla schopna ve videosekvenci detekovat pohybující se objekty a ty následně sledovat po celou dobu jejich výskytu ve scéně. Takováto aplikace může tvořit základ pro ještě sofistikovanější software s pokročilejšími funkcemi. Pokud dokážeme detekovat pohybující se objekty, vzájemně je rozlišit a stanovit historii jejich pohybu, můžeme následně tento pohyb analyzovat. Ze znalosti trajektorie sledovaného objektu můžeme například predikovat následující stav scény. Na základě statistické analýzy pohybu dokážeme definovat běžný pohyb objektu scénou. Jakákoliv odchylka od obvyklého modelu chování je pak snadno odhalena. Tato funkce nalezne uplatnění v mnoha aplikacích. Dokážeme spočítat průchody lidí v různých směrech, detekovat pohyb v protisměru nebo třeba odhalit netypické (podezřelé) chování zákazníka v obchodě. Jistě bychom nalezli mnohá další praktická využití. Aplikace se však omezuje na detekování pohybujících se objektů (lidí), jejich trasování a dále naznačuje možnosti využít tohoto k jednoduchým úkolům jako je získání základních statistik pohybu objektů ve scéně. Konkrétně implementovanou funkcí je počítání osob. Sledovanými objekty jsou zde lidé (jednotlivé osoby), proto dochází v práci v tomto smyslu k volnému zaměňování obecnějšího pojmu objekt a osoba.

3.1 Nástroje a prostředky

Aplikace byla vyvíjena a testována na notebooku Acer Extensa 5220, s procesorem Intel Core 2 Duo 2,2 Ghz, 2 GB RAM, integrovanou grafickou kartou Intel X3100, operačním systémem Windows 8.1 a vývojovým prostředím Microsoft Visual Studio 2012. Stanice s touto konfigurací hardware již evidentně nepatří k nejvýkonnějším a tato skutečnost v některých případech ovlivnila volbu použitých metod.

Pro implementaci aplikace byla využita otevřená multiplatformní knihovna OpenCV ve verzi 2.4.8. Jde o široce rozšířenou knihovnu určenou pro práci z obrazem v reálném čase. Knihovna je vydávána pod BSD licencí umožňující volné využití v akademické i komerční sféře. Samotná knihovna je napsána v jazyce C/C++ a i celá aplikace byla napsána v programovacím jazyce C++. S tímto jazykem jsem pracoval poprvé, ale přesto jsem mu dal přednost před ostatními. Použití „pohodlnějšího“ jazyka jako je C# by znamenalo nutnost použití wrapperu knihovních funkcí (např. OpenCVSharp nebo Emgu). To by mělo za následek lehké zpomalení aplikace. S ohledem na parametry vývojářské stanice jsem tedy raději volil počáteční nepohodlí při vývoji a alespoň částečně vyšší výkon jazyka C++. Pro volbu jazyka C++ hovoří i jeho nezávislost na použitém operačním systému. Zvolený jazyk tak neomezuje možnou přenositelnost vytvořeného algoritmu. Pro úplnost dodávám, že knihovna OpenCV disponuje podporou pro hardwarovou akceleraci CUDA (OpenCL) a podporou nízkoúrovňové knihovny pro zpracovaní medií a dat Intel IPP.



Obrázek 14: Výsledný trasovací algoritmus.

3.2 Algoritmus

Obrázek 14 zobrazuje výsledný algoritmus, který reprezentuje vytvořenou aplikaci. Postupně jsou zde představeny vybrané metody a postupy, které byly použity při implementaci. Základními kroky posloupnosti zpracování snímků mohou být:

- načtení snímku a jeho předzpracování,
- detekce pohybu,
- segmentace objektů,
- přiřazení nalezených objektů k objektům ve sledování.

Jednotlivé dílčí kroky algoritmu jsou blíže popsány v následující části práce.

3.2.1 Předzpracování snímku

Vše začíná načtením snímku. Běžně se snímek před dalším zpracováním upravuje, aby bylo dosaženo lepších výsledků. Většinou jde o jasové normalizace jako je gama korekce a roztažení kontrastu.V aplikaci však není nový snímek nikterak předzpracován, pouze je upravena jeho velikost na 640×480 pixelů. Pokusy o úpravu snímku totiž neprokázaly výrazné zlepšení, proto bylo od předzpracování upuštěno.

3.2.2 Detekce pohybu

Na začátku implementace jsem experimentoval s vlastní implementací detektoru pohybu. S využitím funkcí knihovny OpenCV jsem se pokusil vytvořit detektor založený na výpočtu absolutního rozdílu aktuálního snímku a referenčního snímku modelu pozadí. Před samotným odečtením snímků jsem se pokoušel o různé metody předzpracování. Pokusil jsem se vstupní snímek normalizovat úpravou kontrastu, upravit rozostřením nebo naopak přiostřením. Experimentoval jsem s možnostmi samostatných operací i s jejich vzájemnými variacemi. Nejdříve jsem použil statický referenční snímek pozadí získaný z videosekvence. Referenční snímek zachycoval pouze objekty pozadí a v čase nebyl aktualizován. Tento ve své podstatě naivní přístup fungoval dobře, pokud ve videu nedocházelo ke změnám osvětlení a video mělo dobrou kvalitu. Již malá, pro lidské oko občas téměř nepostřehnutelná změna světelných podmínek v určité oblasti obrazu, způsobovala algoritmu problémy. Částečně se dařilo řešit problém úpravou snímku vhodným předzpracováním a použitím morfologických operací po prahování výsledného snímku odečtu. Celkově byl však výsledek neuspokojivý. Experimentoval jsem tedy i s možnostmi aktualizace referenčního snímku. Byly provedeny pokusy s průměrováním referenčního snímku s různou periodou aktualizace a různou váhou, podle které se nové snímky promítly do modelu pozadí.

Dalším problémem, který jsem se pokusil řešit, byla eliminace stínů v obraze. Při testování algoritmu totiž docházelo nejen k označení siluety postavy, ale byl zahrnut i její

stín. Ten v některých případech způsoboval splynutí dvou siluet v jeden blob a tedy chybnou segmentaci postav. Pokusil jsem se pomocí Cannyho detektoru nalézt hrany v rozdílovém snímku. Odstranit vnější hrany a vnitřní hrany opět vyplnit v jeden blob. Vnější hrany představují okrajové části detekované oblasti a tedy i stíny. Jejich ořezáním se tedy zbavíme i stínů. Vyplněním vnitřních hran by však mělo hlavní „tělo“ objektu zůstat zachováno. Tento postup se mi ale nepodařilo plně realizovat. Možnosti jeho uplatnění by spíše byly u objektů s členitější vnitřní strukturou.

Výsledky takto vytvořeného detektoru pohybu jsem porovnal s výsledky detektoru Mixture of Gaussians implementovaného v knihovně OpenCV. Porovnání bylo pouze orientační a probíhalo na videu z webkamery notebooku. Cílem totiž nebylo samotné srovnání metod (to není předmětem této práce), ale mělo za cíl určit vhodnější metodu s ohledem pro další zpracování. Hlavními kritérii byly rychlost algoritmu, aby se samotná detekce oblastí pohybu co nejméně podílela na celkovém čase výpočtu algoritmu sledování objektů zájmu a samozřejmě přesnost detekce. Nutnost použití dodatečných morfologických operací ve mnou vytvořeném detektoru snižovala rychlost výpočtu jinak rychlejšího mého algoritmu na srovnatelnou rychlost s metodou MOG2 z OpenCV. Sofistikovanější metoda MOG2 však přece jen poskytovala kvalitnější výstup a proto jsem jí dal v následující implementaci přednost.

Obrázek 15: Originální snímek a výsledek detekce pohybu pomocí MOG2.

Ve výsledné aplikaci je pro detekci pohybu použita třída BackgroundSubstractorMOG2 z knihovny OpenCV, implementující algaritmus z [9].Počáteční parametry jsou ponechány v implicitním nastavení (viz dokumentace OpenCV[1]). První parametr určuje délku udržované historie, druhý parametr je prahová hodnota pomáhající přiřadit aktuální pixel k některému z gaussiánů. Pokud nedojde k přiřazení k žádnému z gaussiánů, je vytvořen nový. Počet gaussiánů je stanoven dynamicky pro každý pixel. (Maximum je implicitně 5.) Parametr fTau je nastaven na $0,5$. Tento parametr je prahová hodnota pro detekované stíny. Pixel je označen jako stín, pokud je jeho barva stejná jako pozadí, pouze je tmavší. Hodnota $0,5$ udává, že pokud je pixel více než $2\times$ tmavší než pozadí, není označen jako stín. Dále je pomocí parametru nShadowDetection změněna barva detekovaného stínu na 0, tedy černou. Tím dojde k jeho „smazání“.

[1]Dostupné z http://opencv.org

```
BackgroundSubtractorMOG2 bg(3000, 9, true);
bg.set("fTau", 0.5);
bg.nShadowDetection = 0;
```

Výpis 1: Nastavení detektoru pohybu MOG2.

3.2.3 Segmentace postav

K detekci postav ve snímku se nabízí hned několik metod představených v první části tohoto textu. Mojí původní představou bylo nalezení postavy pomocí metody HOG. Tato metoda poskytuje velmi kvalitní výsledky a je již implementována v knihovně OpenCV. Hned od počátku experimentování se však naplno projevila pravděpodobně její nejvýraznější nevýhoda, kterou je výpočetní náročnost. V kombinaci se slabým výkonem vývojářské stanice byla náročnost metody markantní. V prvotním nastavení trvalo zpracování jednoho snímku 4–6 vteřin. Po úpravách nastavení metody a celého algoritmu aplikace došlo k výraznému zlepšení. Byly změněny parametry metody a prohledávaná oblast byla omezena pouze na oblasti s detekovaným pohybem. V případě většího pohybu osob ve snímku však byla doba detekce stále v řádech vteřin. Pokud má mít algoritmus ambice realtime aplikace, je to nedostačující, proto jsem od původního záměru ustoupil.

Další zkoušenou metodou byl detektor Viola & Jones opět implementovaný v knihovně OpenCV. Jeho výsledky při použití klasifikátu, který je součástí knihovny, nebyly tak dobré jako u metody HOG, čas zpracování byl ale výrazně lepší. I v tomto případě však bylo potřeba omezit prohledávanou oblast pomocí detekce pohybu. Nakonec se ukázalo jako nejvýhodnější použít pro segmentaci postav pouze kvalitní detektor pohybu. Jeho volba byla popsána v předchozí kapitole. Detektor HOG byl za daných podmínek nepoužitelný a metoda Viola & Jones nevykazovala výrazný přínos. Podrobné srovnání metod zde není uvedeno. Nebylo to náplní této práce.

Na obrázku 15 jsou jasně patrné bílé plochy představující oblasti pohybu ve scéně. Tyto objekty přesně korespondují s hledanými postavami osob. A právě tento jednoduchý přístup je využit k segmentaci osob z obrazu. V porovnání s předchozími experimenty došlo k razantnímu zrychlení, protože detekce pohybu stejně již v obou případech probíhala. Očividnou nevýhodou této metody je situace, kdy dojde ke spojení více postav v jednu oblast. Pak se navržený postup dostává do problémů. Snímek pohybujících se objektů získaný detekcí pohybu je zbaven velmi malých objektů. Ty ve většině případu reprezentují šum a oblasti s malým pohybem. Tato filtrace je provedena pomocí morfologické operace Otevření. Otevření je aplikace operací matematické morfologie Eroze a Dilatace v tomto pořadí. Eroze zajistí odstranění detailů a Dilatace opětovné vyplnění a zachování velikosti objektů. Mimo odstranění oblastí, které jsou menší než použitý strukturní element, zajistí operace Otevření i rozdělení objektů spojených pouze úzkými spoji. To napomůže správné segmentaci postav, které jsou detektorem pohybu spojeny v jeden objekt. Následná segmentace je provedena nalezením kontur v připraveném obraze. Je použita funkce z knihovny OpenCV findContours.

```
findContours(foreground_frame, contours, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE)
```

Výpis 2: Funkce pro nalezení kontur.

Kde foreground_frame je upravený výstup detektoru pohybu a parametr contours je vektor nalezených kontur. Kontury jsou zde reprezentované vektory aproximovaných bodů. Použití parametru CV_RETR_EXTERNAL zaručuje, že budou vráceny pouze vnější kontury. Jsou tedy hledány pouze obrysy postav. Poslední parametr určuje použitou metodu aproximace bodů. V tomto případě CV_CHAIN_APPROX_SIMPLE nevrací všechny body, ale pouze krajní body jednotlivých segmentů kontury. Následně jsou odstraněny malé kontury. Jak velké kontury budou ještě odstraněny, volí uživatel nastavením daného parametru v konfiguračním souboru aplikace. Může tak určit nejmenší možnou velikost objektu, který bude sledován. Protože nebyla použita žádná sofistikovaná metoda detekce lidské postavy, je to jedno z kritérií napomáhající správné segmentaci osob. Jelikož sledování není primárně založeno na vzhledu, je nutné vhodnou volbou pomocných parametrů eliminovat objekty, které nejsou lidé. Parametrem pro označení objektu jako osoby je předpokládaná maximální výška postavy ve videu. Všechny větší objekty nejsou považovány za postavu. Další pozorovanou hodnotou je poměr výšky a šířky. Toto omezení vychází z charakteristického vzhledu stojící postavy. Zde stanovenou podmínkou je, že výška postavy nesmí být menší než jeden a půl násobek její šířky. U vzpřímeně stojící postavy by tento poměr byl patrně vyšší, zde je však nutné brát ohled na odlišný tvar těla během jeho pohybu. Při chůzi a běhu se poměr výšky a šířky siluety mění, přesto by neměl být menší než hodnota $1,5$. V opačném případě není nalezená kontura označena jako postava. Takovými objekty mohou být spojené siluety osob, které se vzájemně míjejí nebo kráčí vedle sebe. Dalším typickým případem pro zde testovaná videa je označení osoby i s táhnutým kufrem. V první části obrázku 16 je osoba označena i s kufrem. Nesplňuje tak parametry stanovené pro postavu. V druhé části téhož obrázku je osoba extrahována přesně a splňuje podmínky pro označení objektu jako osoby.

V aplikaci je rozlišení osob a objektů nakonec využito pouze pro vizuální označení. Pro potřeby trasování se detekované bloby nerozlišují. Nepřicházíme tak o cenou informaci vlivem špatného rozlišení postav a jiných objektů. Aplikace tedy dokáže sledovat jakékoli objekty na základě jejich trajektorií, její primární určení je ale sledování osob.

3.2.4 Přiřazení

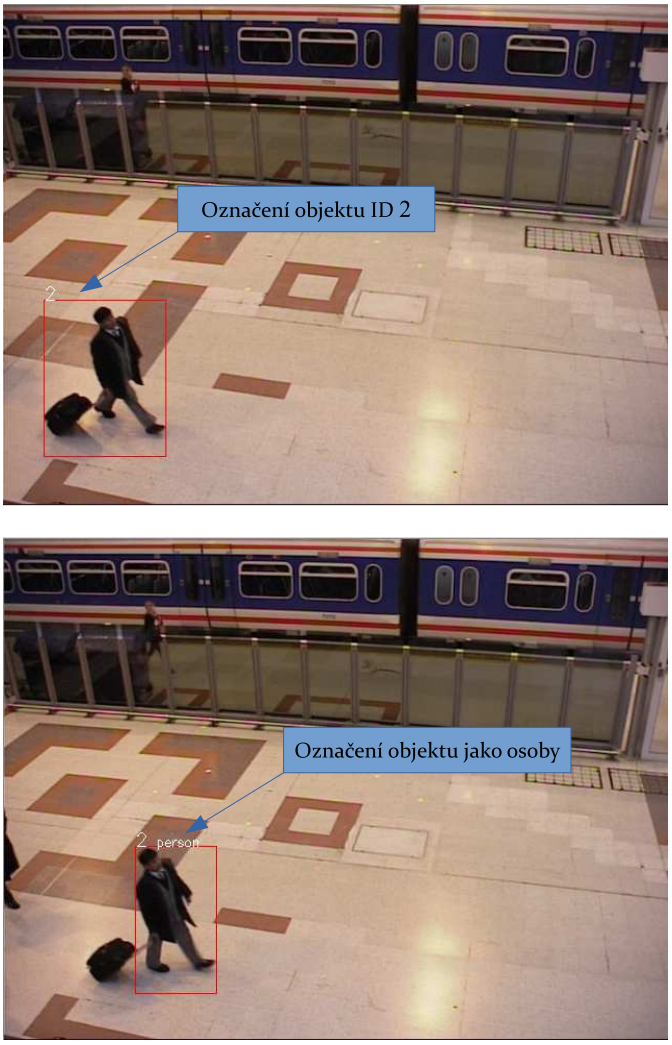
K nalezeným konturám jsou v dalším kroku vypočítány ohraničující boxy. V každém novém snímku detekované objekty jsou dále v algoritmu reprezentovány pouze těmito boxy. Právě k těmto boxům jsou přiřazovány objekty typu TrackingObject, které představují sledované osoby. Určení korespondence osob ve sledování a nově detekovaných postav probíhá na základě výpočtu jejich vzájemné vzdáleností. Jako metrika je použita klasická Euklidovská vzdálenost, definovaná vztahem:

$$D_E = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{29}$$

Kde x_1, y_1 jsou souřadnice predikované polohy sledované osoby a x_2, y_2 jsou souřadnice objektu nalezeného v aktuálně zpracovávaném snímku. Maximální akceptovatelná vzdálenost byla stanovena na hodnotu dvojnásobku šířky příslušného objektu v minulém snímku:

$$D_E \leq 2 \times w_{t-1}. \tag{30}$$

Nejdříve je spočítána vzájemná vzdálenost pro všechny objekty ve sledování a aktuálně detekované bloby. Je sestavena tabulka vzdáleností, ve které jsou shromážděny veškeré vzdálenosti. Nad touto tabulkou následně dochází k vytváření párů ⟨sledovaný vzor - jeho obraz v aktuálním snímku⟩. Přednost při přiřazení mají relace s nejmenší hodnotou vzdálenosti. Každý blob může být pochopitelně přiřazen pouze k jedné sledované postavě. Ukázku vykreslení trajektorie sledovaných osob lze nalézt v příloze **??**.



Obrázek 16: Označení objektu.

3.2.5 Porovnání se vzorem

V ideálním případě je segmentace úplná, tedy všechny objekty zájmu v obraze jsou extrahovány korektně. V komplexní scéně reálného prostředí však dochází vzájemnému překrývání objektů a tím i k částečnému nebo úplnému zakrytí jednoho z objektů. Velmi častým problémem zvoleného přístupu je při částečném překrytí spojení dvou siluet v jeden velký objekt. Pak dojde k nepřesnému přiřazení pouze jednoho objektu a druhý zůstane bez aktualizace. V takovém případě je v aplikaci vypočítána tabulka možných kolizí, která udržuje zjištěné kolizní objekty. Při výpočtu možných kolizí je detekováno překrytí ohraničujících boxů jednotlivých postav a ty jsou pak zaznamenány. Samotné překrytí ohraničujících boxů dvou objektů nemusí znamenat jejich skutečné překrytí ve scéně, je to však podmínka nutná, a tedy k detekci možného překrytí dostačující.

V situaci, kdy některému z objektů ve sledování není přiřazen jeho obraz v aktuálním snímku, je vytvořena tabulka kolizí. Pokud uvažovaný objekt není v kolizi s jiným objektem, je mu nastavena nová pozice pomocí predikce Kalmanova filtru. V opačném případě je zrušeno nastavení nových poloh všech již přiřazených kolizních objektů. V dalším kroku se program pokusí nalézt nové polohy všech dotčených kolizních objektů pomocí srovnání se vzorem. Každý sledovaný objekt si udržuje svůj obraz (vzor) získaný při poslední potvrzeném přiřazení. Poslední známý obraz objektu je porovnán s oblastí předpokládaného výskytu objektu v aktuálním snímku. Aby nemuselo docházet k prohledávání celého aktuálního obrazu a snížila se tak výpočetní náročnost hledání vzoru, je predikcí Kalmanova filtru získána pravděpodobná poloha objektu. Tato oblast snímku je lehce rozšířena, zejména ve směru dosavadního pohybu. A teprve v této oblasti je vzor hledán. Při shodě překračující stanovenou prahovou hodnotu je objektu nastavena nová poloha na pozici největší shody. V opačném případě přichází ke slovu opět Kalmanův filtr. Prahování výsledků maximální shody se ukázalo jako nezbytnost. Pokud bychom automaticky použili souřadnice maximální shody, dojdeme v mnoha případech k horšímu výsledku než by nám poskytl samotný Kalmanův filtr. Narážíme zde na hlavní problém metody porovnání se vzorem, kterým je nepřesnost vzoru. Pohybem objektu ve scéně může docházet ke změně jejich vzhledu. Může například dojít k situaci, kdy je objekt několik snímků zakryt překážkou a poté se opět objeví. Po dobu zakrytí je sledovaný objekt veden pouze Kalmanovým filtrem a jeho udržovaný vzhled se neobnovuje. V okamžiku opětovného výskytu se již reálný a udržovaný vzhled liší. To může způsobit chybné označení polohy. Přes všechna úskalí však měla implementace metody do aplikace výrazně pozitivní vliv na přesnost určování aktuální pozice v případě částečně se překrývajících objektů.

Porovnání se vzorem je provedeno pomocí funkce OpenCV.

```
void matchTemplate(InputArray image, InputArray templ, OutputArray result, int method)
```

Výpis 3: Funkce pro srovnání se vzorem.

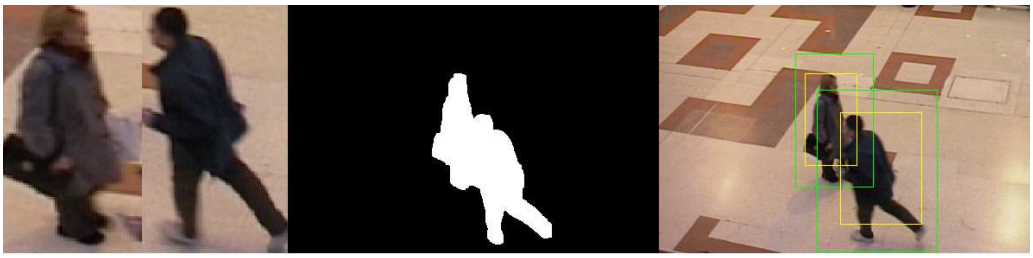
Účely parametrů funkce jsou zřejmé z jejich názvu. Metodu porovnání použitou v apli-

kaci lze popsat následujícím vzorcem:

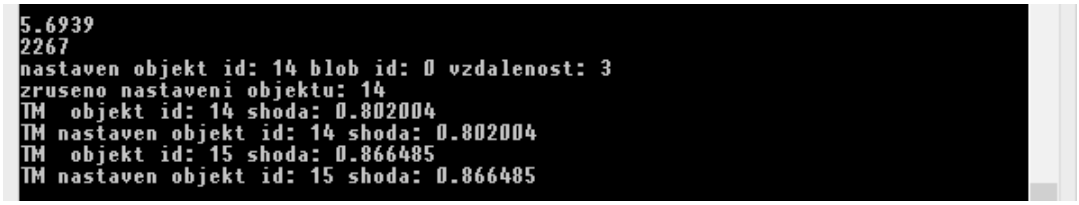
$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}. \tag{31}$$

Kde I je prohledávaný obraz a T je vzor. (Více o implementaci funkce Template matching lze nalézt v dokumentaci OpenCV[2].)

Obrázek 17 ilustruje implementaci metody Template Matching v algoritmu. V sekcích a, b jsou hledané vzory. Ty jsou aktualizovány při každém pozitivním přiřazení. V sekci c je patrné splynutí dvou siluet v jeden blob. Na zde uvedeném obrázku nejsou hledané vzory a prohledávaný obraz ve zkutečném poměru. V tuto chvíli je nejdříve přiřazen jediný nalezený objekt k jedné ze sledovaných postav. V dalším kroku algoritmus zjistí, že druhá postava nebyla přiřazena a detekuje kolizi (překrytí postav). Zruší tedy původní přiřazení první postavy a snaží se nalézt jejich pozici pomocí porovnání se vzorem. V části d jsou zelenou barvou ohraničeny prohledávané oblasti. Žluté obdélníky pak označují oblasti s nejvyšší shodou. Celou proceduru dokládá obrázek 18, kde je zachycen výpis z programu. Jsou zde vypsány i hodnoty nalezených maxim.



Obrázek 17: Porovnání se vzorem.



Obrázek 18: Výpis z programu.

3.2.6 Kalmanův filtr

V aplikaci není využita predikční schopnost filtru pro filtrování naměřených hodnot. Odhad filtru je použit pouze pro určení nové pozice sledovaného objektu v situaci, kdy se

[2]Dostupné z http://opencv.org

algoritmu nepodařilo určit novou pozici jiným způsobem. K tomu dojde zejména pokud se dvě osoby míjejí. Při překrytí osob se algoritmus nejdříve snaží určit novou polohu pomocí metody srovnání se vzorem (viz předchozí kapitola 3.2.5). V případě neúspěchu nastaví novou polohu sledované postavy predikcí Kalmanova filtru. Každý sledovaný objekt udržuje svoji vlastní instanci Kalmanova filtru implementovaného knihovnou OpenCV. Původně jsem zamýšlel využít filtr nejen pro predikci polohy, ale i k predikci velikosti siluety v následujícím snímku. Jelikož nakonec není velikost při přiřazení porovnávána, není ani potřeba predikovat velikost filtrem. Při inicializaci nového objektu je vytvořena i nová instance Kalmanova filtru, která je společná pro predikci polohy ve směru osy x i y. Výpis programu 4 zachycuje nastavení Kalmanova filtru při jeho vytvoření.

```
KF = new KalmanFilter(4, 2, 0);
KF->transitionMatrix = *(Mat_<float>(4, 4) << 1, 0, 1, 0,
      0, 1, 0, 1,
      0, 0, 1, 0,
      0, 0, 0, 1);
KF->statePre.at<float>(2) = 0;
KF->statePre.at<float>(3) = 0;
 setIdentity (KF->measurementMatrix);
 setIdentity (KF->processNoiseCov, Scalar::all(1e-4));
 setIdentity (KF->measurementNoiseCov, Scalar::all(1e-1));
 setIdentity (KF->errorCovPost, Scalar::all(.1));
```

Výpis 4: Inicializace Kalmanova filtru.

3.2.7 Vytvoření a zrušení objektu

Pokud algoritmus nedokáže určit příslušnost detekované postavy k žádnému dosud zaznamenanému objektu, dojde k vytvoření nové instance typu TrackingObject. Nově vytvořenému záznamu je přiděleno ID a je zaveden do seznamu sledovaných objektů. Aby nedocházelo k falešným detekcím, je každý nový objekt nejprve označen jako dočasný. S dočasnými objekty je v aplikaci dále nakládáno stejně jako se stálými. Potvrzené objekty mají pouze přednost při přiřazení. Dočasný objekt je potvrzen na stálý, pokud dojde k jeho opakovanému výskytu.Při inicializaci nového objektu je vedle ostatních parametrů nastavena i položka temp na hodnotu 10, která slouží ke kontrole počtu výskytů daného objektu. Každý dočasný objekt je pak od své první detekce sledován. Pokaždé, kdy dojde k pozitivní detekci, je objektu snížena hodnota položky temp. V okamžiku, kdy je hodnota temp rovna nule, je objekt potvrzen na stálý. Algoritmus tedy pro potvrzení objektu vyžaduje 10 pozitivních přiřazení. Přiřazení však nemusí proběhnout v 10 po sobě jdoucích snímcích.

Důvodem procesu vytvoření nového objektu typu TrackingObject je vždy detekce nového pohybujícího se objektu ve sledované scéně. K tomu může dojít z několika příčin:

- nový objekt vstoupil do záběru,
- sledovaný objekt nebyl přiřazen,

- rozpad sledovaného objektu na více částí.

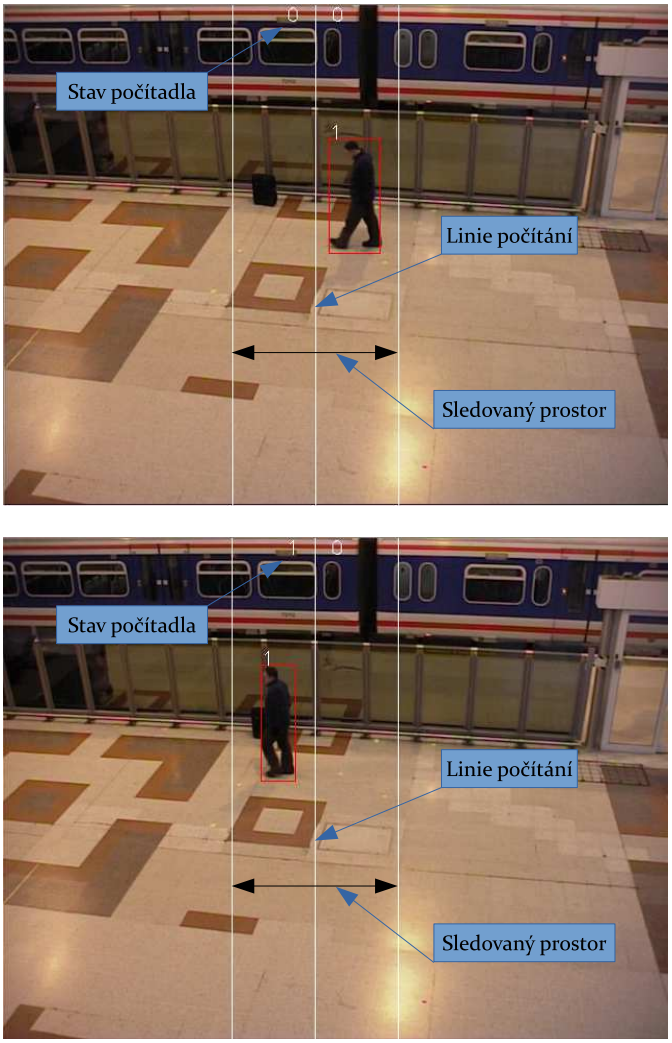
První případ je zcela zřejmý. K druhému případu dojde, pokud nebyla splněna podmínka přiřazení. To znamená, že vzdálenost nové pozice v obraze a předpokládané pozice je příliš velká. K této situaci může dojít například náhlým zrychlením pohybu objektu. Ten mezi jednotlivými snímky urazí větší vzdálenost, než vzdálenost, kterou akceptuje algoritmus na základě predikce Kalmanova filtru. Stejný dopad by mohla mít i prudká změna směru. Algoritmus pak neidentifikuje nalezený objekt a chybně jej vyhodnotí jako nový. Posledním uvedeným případem, který vede k vytvoření nových instancí je rozpad sledovaného objektu na více částí. K tomu může dojít špatnou segmentací. Ve většině případů je takový stav pouze dočasný. Dokáže jej tedy eliminovat právě podmínka potvrzeného výskytu. Dalším možným scénářem je současný vstup dvou osob do záběru. Pokud se postavy částečně překrývají a až poté se jejich trajektorie rozdělí, dojde nejdříve k vytvoření pouze jednoho objektu. Až po rozdělení se inicializuje i druhý objekt.

Každý objekt také obsahuje položku timeStamp, ve které se udržuje počet po sobě jdoucích snímků, ve kterých se daný objekt nevyskytl. Pokud položka timeStamp přesáhne 20 je objekt zahozen. Tím je zajištěno smazání již neaktuálních objektů. Zde uvedené hodnoty položek temp a timeStamp byly nastaveny empiricky. U jiného typu a kvality záznamu by se pravděpodobně lišily.

3.3 Počítání osob

Počítání osob je v aplikaci realizováno pouze na základě trasovacího algoritmu. Aplikace je schopna sledovat postavy po celou dobu jejich výskytu v obraze. To znamená, že známe ucelenou historii pohybu každé osoby. Stačí stanovit pomyslnou linii a poté již jen kontrolovat, které osoby tuto hranici během svého pohybu překročily. Získáme tím statistiky o počtu procházejících osob i s rozlišením směru jejich průchodu. Vše potřebné tedy zajistí trasovací algoritmus. Nastavení sledované linie se provede pomocí konfiguračního souboru před spuštěním aplikace. Kontrolovaná hranice by neměla ležet příliš blízko okrajů. Při vstupu nového objektu do scény totiž může určitý krátký čas trvat, než se objekt „ustálí“ a má relevantní historii.

Každá osoba, která vstoupí do sledované oblasti, je zaznamenána. Tuto situaci ilustruje obrázek 19 (první část). Můžeme zde pozorovat muže, který z pravé strany vstoupil do sledovaného prostoru. Aplikace si v tuto chvíli uloží jeho ID. Muž pokračuje dále a vstoupí do levé části sledované oblasti (druhá část obrázku). Aplikace opět uloží jeho ID, tentokrát pro levou oblast. V následujícím kroku algoritmus podle ID zjistí, že muž již byl v pravé části hlídaného prostoru a musel tak protnout linii počítání. Je tedy zvýšena hodnota příslušného počítadla. Uložení ID zabraňuje vícenásobné inkrementaci počítadla jednou osobou. Každá osoba je proto během svého pohybu v hlídaném prostoru započítána pouze jednou. Po opuštění sledovaného prostoru je záznam smazán a osoba může být při novém průchodu opět započítána. Zbývá jen dodat, že šířka sledované oblasti byla nastavena na 80 pixelů z obou stran linie počítání. (Na rozdíl od obr.19 není aplikací ve skutečnosti hranice sledované oblasti vizuálně zobrazena.)

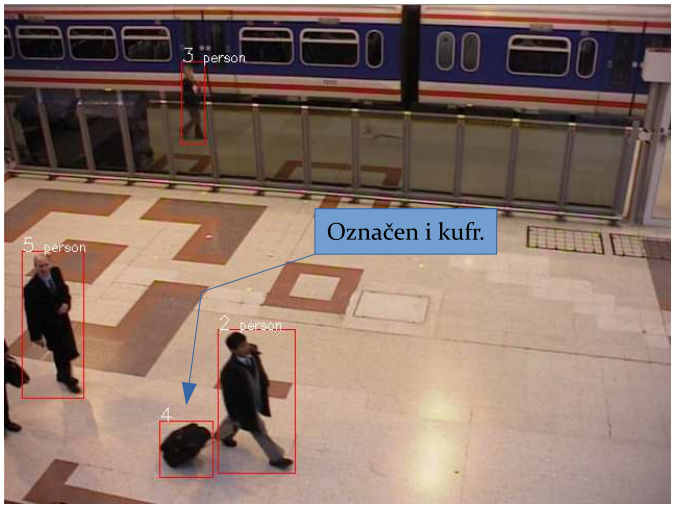


Obrázek 19: Počítání osob.

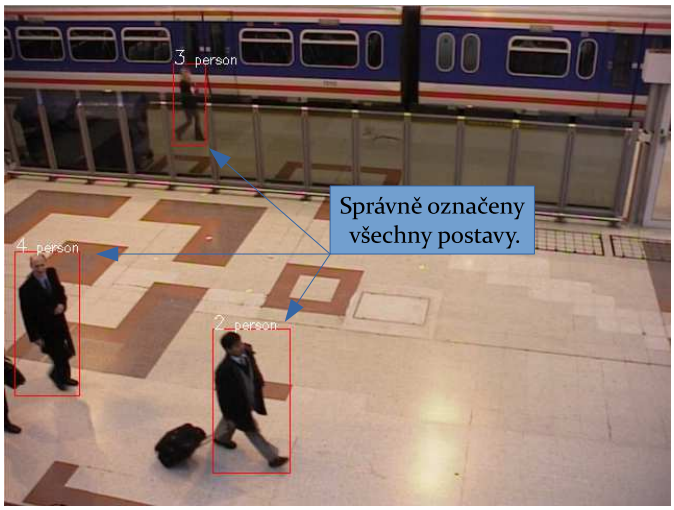
3.4 Nastavení parametrů

Program byl vytvořen jako konzolová aplikace. Veškeré nastavování parametrů probíhá prostřednictvím konfiguračního textového souboru. Jeho podrobnější popis lze nalézt v souboru „readme.txt“, který je přiložen k této práci. Zde se jen pozastavíme u nejdůležitějšího parametru, který výrazně ovlivňuje přesnost detekce a následné počítání osob. Jeho volba může mýt zásadní vliv na dosažené výsledky aplikace. Tímto parametrem je minHeight, který definuje minimální velikost pro sledované objekty. Obrázek 20 ilustruje vliv tohoto parametru na výsledky sledování osob.

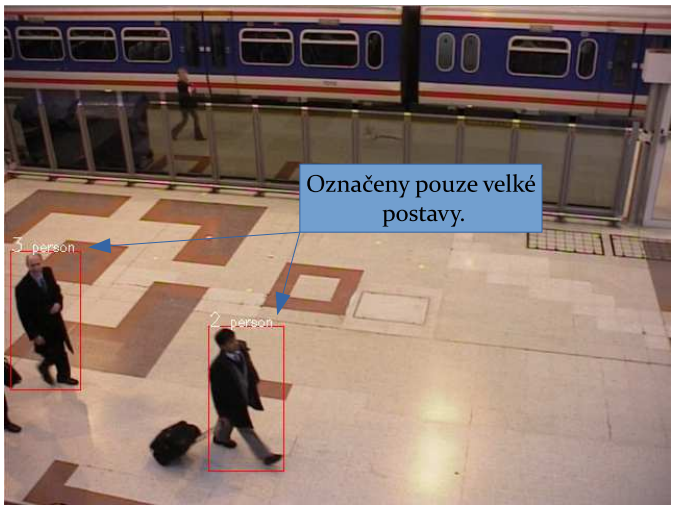
minHeight=40



minHeight=65



minHeight=80



Obrázek 20: Vliv parametru minHeight na správnost detekce.

U prvního snímku byl nastaven parametr minHeight na 40 pixelů. Díky tomu sleduje program i pohyb malé postavy v pozadí snímku. Nízká hodnota parametru má ale za následek i označení kufru jako samostatného objektu ke sledování. Kufr sice není označen jako osoba, jeho pohyb je ale sledován stejným způsobem jako ostatní postavy. Druhý snímek zachycuje stejnou situaci se správně zvolenou úrovní minHeight. Jsou sledovány všechny postavy v obraze a kufr přitom zůstal neoznačen. Třetí snímek opět zachycuje stejnou situaci, tentokrát s příliš vysokou hodnotou sledovaného parametru. Jsou sledovány pouze výrazné postavy a ostatní objekty jsou ignorovány.

3.5 Návrh algoritmu pro detekci pádu

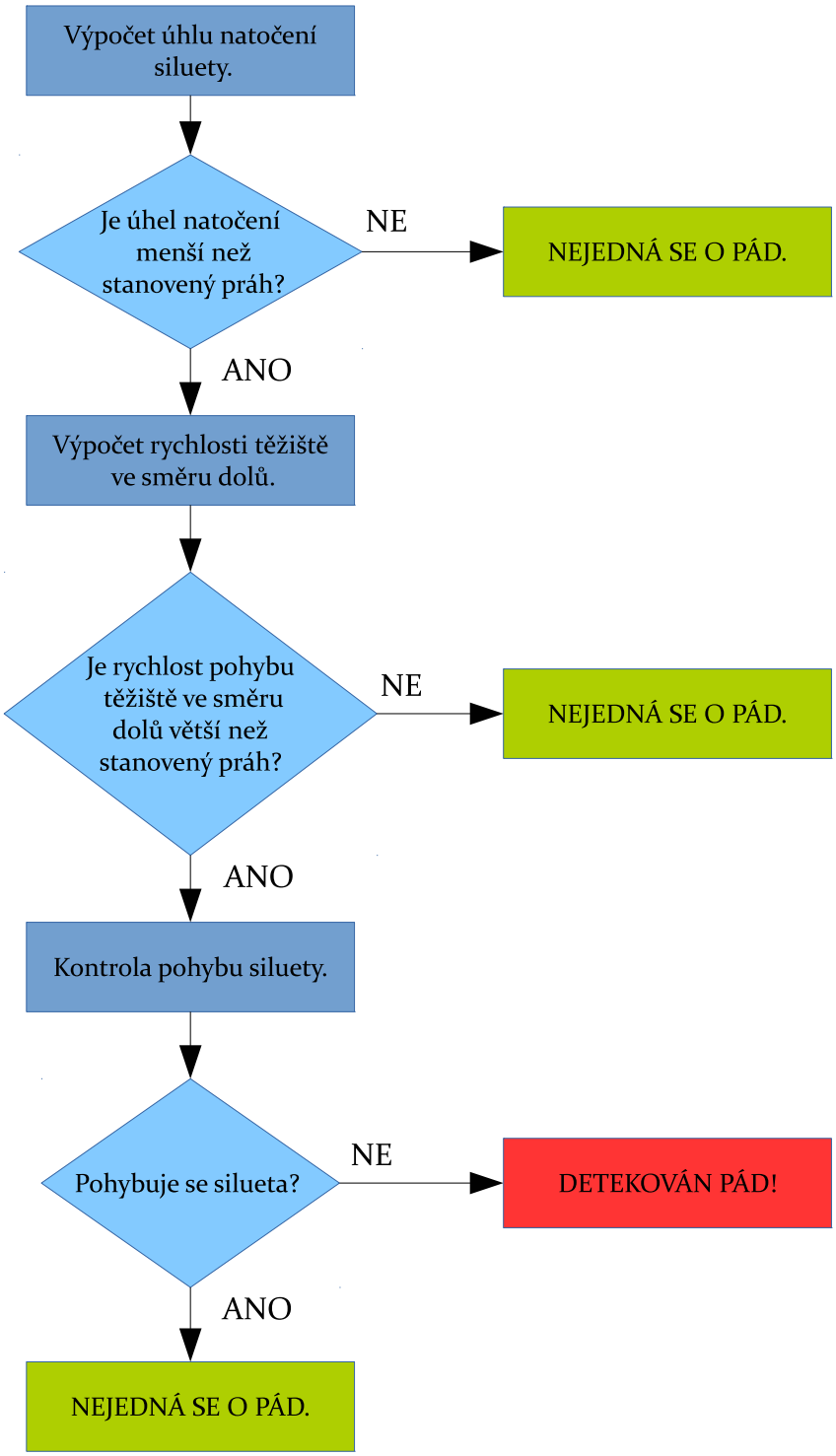
Z metod uvedených v kapitole 2.5 se pro mnou vytvořenou aplikaci nejvíce hodí detekce založená na vlastnostech extrahované siluety. Analýzou tvaru siluety a její historií pohybu lze získat potřebné příznaky pro klasifikátor detekce pádu. Základními příznaky by mohly být:

- úhel natočení,
- rychlost pohybu těžiště ve směru dolů,
- aktuální nehybnost.

Hlavní osa stojící postavy za běžných okolností směřuje kolmo. Její úhel natočení je tedy 90 °. V případě pádu se úhel těla a vodorovné osy snižuje. Vhodnou volbou prahové hodnoty úhlu natočení můžeme získat první stupeň klasifikace. Pro zjištění úhlu lze použít funkci fitEllipse z knihovny OpenCV. Funkce vrací objekt typu RotatedRect s požadovaným parametrem angle.

V druhém kroku můžeme zkontrolovat rychlost těžiště nebo horní části siluety. Důležitým ukazatelem je i směr pohybu sledovaného bodu na siluetě. V případě pádu bude s největší pravděpodobností těžiště směřovat dolů. Tyto parametry lze získat analýzou trajektorie pohybu v blízké minulosti.

Doplňující podmínkou může být nehybnost, respektive malý pohyb, následující po pádu. Vycházíme z předpokladu, že osoba, která upadla a má problémy, se nehýbe nebo jen omezeně. Pokud budeme akceptovat i malý pohyb, eliminujeme i chyby způsobené měřením. Grafické znázornění navrženého algoritmu je na obrázku 21.



Obrázek 21: Navržený algoritmus pro detekci pádu.

4 Testování a výsledky

4.1 Testovací data

K testování schopností aplikace byl zvolen dataset PETS 2006 (obrázek 22). Tento dataset je určen pro vývoj a testování aplikací, které se zabývají sledováním lidí ve videosekvencích. Sloužil pro účely mezinárodního semináře „Performance Evaluation of Tracking and Survaillance“. Seminář je každoročně pořádán institutem IEEE a jeho náplní je vývoj v oblasti automatizovaného sledovaní a porozumění lidského chování v dynamických obrazových datech. Použitý soubor dat obsahuje záznamy z kamer pořízené v prostorech vlakového nádraží při běžném provozu. Snímanou scénou procházejí cestující v různých směrech a různou rychlostí. Tím, jak se vzájemně míjejí, dochází k jejich překrývání, a tedy i k vzájemnému křížení jejich trajektorií. Tento scénář je tedy vhodnou volbou k testování navrženého trasovacího algoritmu. Pořízené videozáznamy jsou uloženy jako sekvence jednotlivých snímků ve formátu JPEG s rozlišením 720×576 pixelů.

Pro doplnění hlavního testovacího souboru byl určen soubor obdobný předešlému. Jde opět o dataset semináře PETS, tentokrát však z roku 2009. Snímanou scénou zde nejsou vnitřní prostory nádraží ale venkovní prostory (obrázek 23). Záznam vznikl na univerzitě v Readingu právě pro účely testování trasovacích algoritmů. Opět zde můžeme pozorovat chodce, kteří kráčejí v různých směrech, vzájemně se míjejí a tentokrát i úmyslně proplétají a mění směr.

Obrázek 22: Ukázka testovacích dat - PETS2006

Obrázek 23: Ukázka testovacích dat - PETS2009

4.2 Testování vybraných snímků

V následující sekci jsou stručně shrnuty výsledky počítání osob ve vybraných sekvencích snímků z datasetu PETS 2006. Z uvedeného souboru jsou vybrány pouze záznamy kamer z bočního pohledu, které jsou k testování funkce počítání nejvhodnější. Považuji za nutné zdůraznit, že při jiném nastavení parametrů se výsledky mohou (budou) lišit. Nejen hodnota již probíraného parametru minHeight, ale i umístění linie počítání má zásadní vliv na dosažené výsledky.

Výsledky pro jednotlivá videa jsou zobrazeny v tabulkách, kde řádky označeny $\leftarrow$, $\rightarrow$ reprezentují směr průchodu zleva doprava, respektive zprava doleva.

video: S1-T1-C
minHeight: 65
linie počítání: 190

	započítáno	skutečně prošlo	chyby	úspěšnost
$\rightarrow$	18	20	2	–
$\leftarrow$	15	17	1	–
celkem	–	–	3	89,19%

Tabulka 1: Výsledky pro video S1-T1-C.

Poznámka: Chyby v počítání osob byly způsobené špatnou segmentací postav. Ke dvěma chybám ve směru vpravo došlo v horní části scény, kde jsou postavy poměrně malé. Zejména v jednom případě by mohlo být překrytí dvou postav špatně rozlišitelné i pro nepozorného lidského pozorovatele. Ve směru vlevo způsobila chybu skupinka čtyř žen, kterou program rozdělil pouze na dvě části.

video: S2-T3-C
minHeight: 65
linie počítání: 190

	započítáno	skutečně prošlo	chyby	úspěšnost
$\rightarrow$	7	19	12	–
$\leftarrow$	21	23	2	–
celkem	–	–	14	66,67%

Tabulka 2: Výsledky pro video S2-T3-C.

Poznámka: U tohoto videa byla způsobena většina chyb nezapočítáním malých postav v horní části scény. Tyto postavy se navíc pohybovaly za odstavenými kontejnery, čímž byla jejich detekce takřka znemožněna. Je tedy otázkou, zda za takovýchto podmínek považovat výsledek za špatný. Možná by bylo vhodnější rezignovat na snahu započítat jakýkoliv, a tedy i vzdálený průchod, a nastavit algoritmus na počítání osob pouze v hlavní části obrazu. V tomto videu je také zřetelně vidět problém s aktualizací modelu pozadí při změně scény (obr. 24).

video: S3-T7-A
minHeight: 65
linie počítání: 160

	započítáno	skutečně prošlo	chyby	úspěšnost
→	8	9	1	–
←	7	8	2	–
celkem	–	–	3	88,24%

Tabulka 3: Výsledky pro video S3-T7-A.

Poznámka: Za povšimnutí stojí řádek pro počet osob ve směru vlevo. Rozdíl osob započítaných aplikací a skutečný počet osob je pouze 1. Počet chyb počítání je však uveden 2. Důvodem je kufr tažený započítanou osobou. Ten byl aplikací veden jako samostatný objekt a způsobil tak chybné navýšení počitadla. Řešením této chyby by bylo zvýšení hodnoty parametru minHeight na 80 pixelů. To by byla dostatečná hodnota, která by eliminovala veškeré malé objekty, jako jsou právě kufry atd. Pravděpodobně by pak ale aplikace nebyla schopna sledovat pohyb ani malých dětí a již zmíněných malých postav v horní části scény.

Další pozorovatelný problém v tomto videu způsobuje opět model pozadí. Postava, která se chvíli nepohybuje a stojí na místě, po chvíli „splyne“ s pozadím. K opětovnému označení postavy dojde až po jejím dalším pohybu. Nové označení postavy nepůsobí potíže při počítání, jde ale o chybu v trasovaní. Řešením je úprava nastavení detektoru pohybu.

video: S4-T5-A
minHeight: 65
linie počítání: 160

	započítáno	skutečně prošlo	chyby	úspěšnost
→	7	7	0	–
←	10	11	1	–
celkem	–	–	1	94,12%

Tabulka 4: Výsledky pro video S4-T5-A.

Poznámka: Stejné potíže, které již byly zmíněny. Pohyb posuvných dveří v pravém horním rohu způsobuje krátkodobé chybné označení neexistujících objektů.

video: S5-T1-G
minHeight: 65
linie počítání: 160

	započítáno	skutečně prošlo	chyby	úspěšnost
→	15	17	2	–
←	7	15	8	–
celkem	–	–	10	66,67%

Tabulka 5: Výsledky pro video S5-T1-G.

Poznámka: U tohoto videa aplikace chybovala opět u malých postav v horní části a se skupinou osob.

video: S6-T3-H
minHeight: 65
linie počítání: 160

	započítáno	skutečně prošlo	chyby	úspěšnost
→	9	10	1	–
←	6	7	1	–
celkem	–	–	2	88,24%

Tabulka 6: Výsledky pro video S6-T3-H.

Poznámka: Jde o video bez většího pohybu osob.

video: S7-T6-B
minHeight: 65
linie počítání: 160

	započítáno	skutečně prošlo	chyby	úspěšnost
→	18	25	7	–
←	21	21	0	–
celkem	–	–	7	84,78%

Tabulka 7: Výsledky pro video S7-T6-B.

Poznámka: V některých chvílích je ve scéně pohyb více osob, které vstupují do scény samostatně a aplikace proto funguje bezchybně. Jinak se vyskytly stejné problémy jako v předchozích videích.

4.3 Testování ostatních videozáznamů

Dataset PESTS 2009 obsahuje sekvence snímků scény pořízené z různých vzdáleností a úhlů pohledu. Následkem toho jsou parametry výsledných videozáznamů rozdílné. Aplikace však byla již od počátku vytvářena pro vybraná videa z datasetu PETS 2006. Při závěrečném testování se ukázalo, že některé parametry videa a snímané scény souboru PETS 2006 se částečně promítly i do nastavení algoritmu. Příkladem může být nastavení

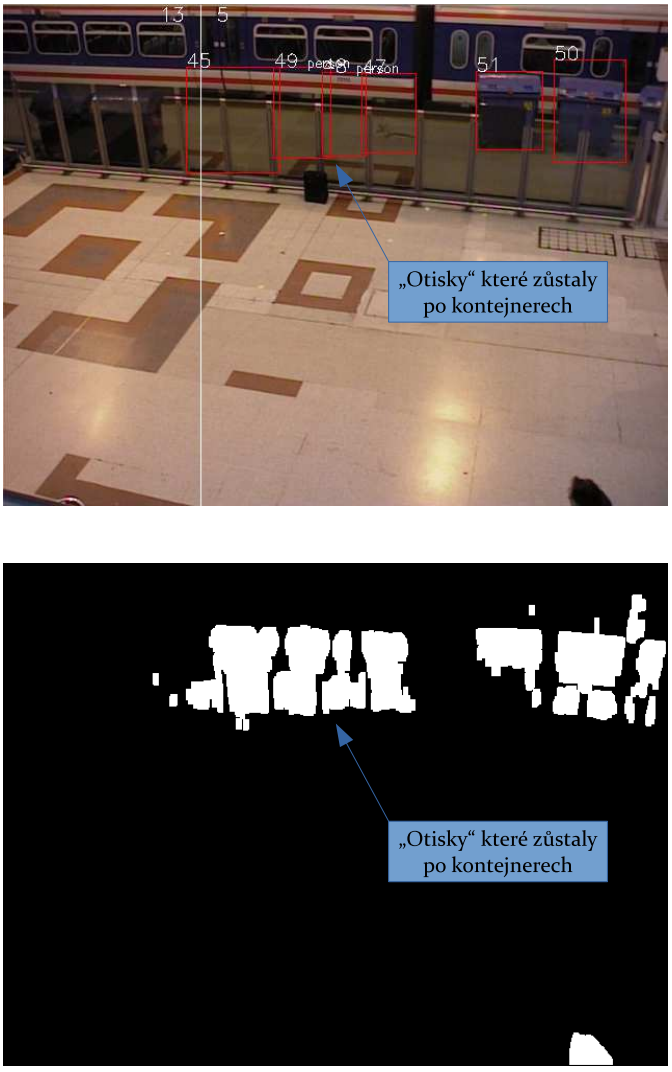
detektoru pohybu, Kalmanova filtru a podmínky korespondence objektů. Bez menších či větších úprav parametrů algoritmu je tak aplikace u některých odlišných videí nepoužitelná. Protože jednotlivá videa vyžadují lehce odlišné nastavení a dosažení univerzálního nastavení je v reálu v podstatě nemožné, od rozsáhlejšího testovaní jiných souborů jsem upustil. Zájemce může zhodnotit funkci algoritmu u odlišného záznamu shlédnutím přiloženého videosouboru *PETS2009.avi*. Zlepšení funkce trasování oproti obecnému nastavení zde bylo dosaženo pouze zakomentováním jednoho řádku kódu aplikace. Pro další zlepšení by bylo třeba upravit nastavení Kalmanova filtru. Tuto možnost však běžný uživatel nemá, proto výsledek prezentuji touto formou. U jiného videa by se rozsah nutných změn pravděpodobně lišil.

4.4 Hodnocení

K hlavnímu testování byly použity sekvence snímků z datasetu PETS 2006 (viz kap. 4.1). Prvotním krokem bylo otestovaní nastavení trasovacího algoritmu. Ten ve své původní podobě neobsahoval funkci porovnání se vzorem. Veškeré situace, kdy nebyl sledovaný objekt v novém snímku identifikován, byly řešeny pouze pomocí predikce Kalmanova filtru. V některých případech však nebylo sledování objektů pouze pomocí filtru přesné. Problém se projevoval zejména při delším překrytí osob. Pokud osoby kráčí rychle proti sobě, je predikce polohy filtrem dostačující informací k rozlišení osob. Může dojít ke krátkodobé záměně ID osob, v další fázi však dojde k nápravě a správnému označení. Náročnější situaci představují osoby, které kráčí směrem k sobě a u sebe se zastaví např. ke krátkému rozhovoru. Rozlišení osob pouze na základě jejich minulého a předpokládaného pohybu může v tomto případě způsobit jejich záměnu, se kterou si již algoritmus nedokáže poradit. Snahou o řešení tohoto problému bylo doplnění algoritmu o metodu porovnání se vzorem (kap.3.2.5). Použitím této metody došlo k poměrně výraznému zvýšení robustnosti algoritmu vůči podobným schématům chování. Nevýhodou je zvýšení výpočetní náročnosti, které je patrné zejména při křížení více původně samostatných objektů. (V takovýchto případech klesá rychlost aplikace k cca 4 fps.) I tento postup však může selhávat v patrně nejnáročnější situaci, kterou je spojení dvou osob a jejich následný pohyb stejným směrem. Osoba, která je zakryta, nemůže být srovnáním se vzorem nalezena a je vedena pouze pomocí predikce Kalmanova filtru. Predikce filtru je však vlivem absence korekce stále více nepřesná a pokud nedojde do určité doby k potvrzení výskytu objektu ve scéně, je nakonec objekt „zahozen“.

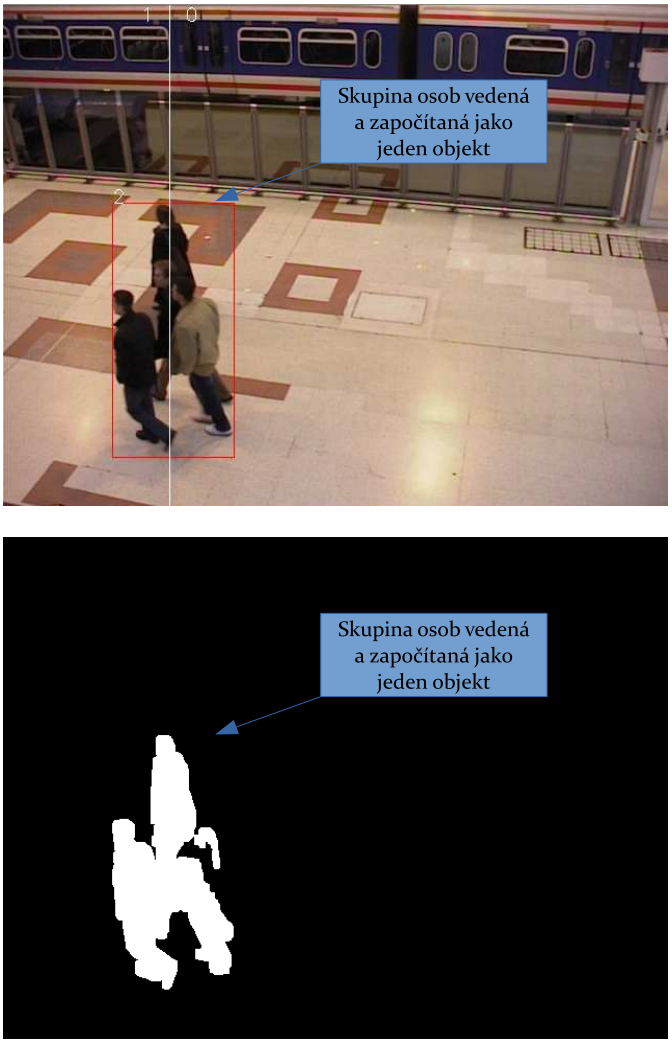
Testování také potvrdilo slabinu zvoleného přístupu k segmentaci postav. Přesnost detekce se odvíjí od přesnosti a spolehlivosti detektoru pohybu. Obrázek 24 ukazuje typický problém se zahrnutím objektů popředí do modelu pozadí a jejich následný pohyb. Ve scéně jsou umístěny kontejnery, které jsou v průběhu videa odstraněny. Jelikož byly začleněny do vytvořeného modelu pozadí, je na jejich místě po jejich odstranění detekován pohyb. Následně určitou dobu trvá než se model aktualizuje. Tuto dobu můžeme ovlivnit nastavením detektoru. Zvolíme-li rychlou aktualizaci, nebudou osoby, které se zastavily a chvíli se nehýbou, detekovány. Je proto nutné nalézt vhodný kompromis. Stejný problém jako s kontejnery by nastal i v případě, že by se rozjel vlak, který kamera zabírá v horní části scény. V celé oblasti by byl detekován pohyb a došlo by tím k znehod-

nocení výstupu detektoru. Možným řešením by bylo, že v případě detekce velké oblasti pohybu by na to algoritmus zareagoval adekvátní odezvou. V případě označení plochy, která se rozměrově nebo tvarově vymyká běžnému pohybu, bychom například mohli resetovat model pozadí novým výpočtem.



Obrázek 24: Změna pozadí.

Další problém, který způsobuje segmentace založená na detekci pohybu, je spojování více osob do jednoho objektu. Jak již zde bylo několikrát uvedeno, dojde při překrytí postav ke spojení jejich siluet v jeden blob. Tento problém jsem se snažil vyřešit použitím detektoru Viola & Jones pro horní část těla. V objektech, které nesplňovaly podmínky pro samostatně stojící osobu, jsem se snažil zmíněným detektorem nalézt horní partie lidské postavy. Tento pokus však (možná vinou použitého deskriptoru) neměl dobré výsledky. Nejlépe si vedla metoda, kdy jsem se snažil do spojených siluet vepsat siluetu uměle

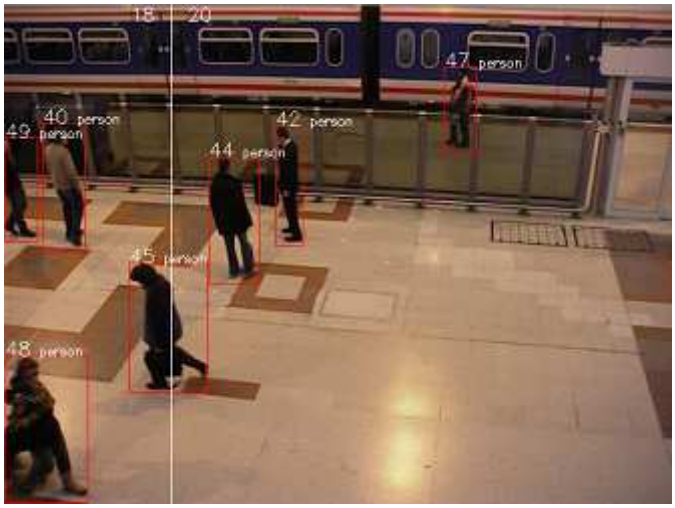


Obrázek 25: Skupina osob.

vytvořenou. Metoda měla poměrně dobré výsledky, byla však velmi citlivá na velikost použitého vzoru. Velikost jednotlivých postav v použitých videosekvencích se totiž liší. Pokusy o dynamickou změnu velikosti podle průměrné velikosti postavy v dané oblasti snímku pak tuto metodu komplikovaly a nepodařilo se mi ji přivést do úspěšného konce. Kromě problémů se správnou velikostí vzoru docházelo také k občasným falešným pozitivním detekcím. Metoda se totiž snažila rozčlenit nejen skupiny osob, ale i velké objekty jako jsou vozíky, atd. Od segmentování spojených postav jsem tedy nakonec upustil. Důsledkem toho jsou osoby, které vstoupí do záběru jako skupina, vedeny jako jeden velký objekt a v případě počítání osob započítány pouze jako jedna osoba (obr. 25). Nicméně osoby, které vstoupí do záběru samostatně a až posléze dojde k jejich spojení, jsou ve většině případů algoritmem zpracovány korektně. Díky použití Kalmanova filtru a metody

porovnání se vzorem se aplikace s dočasným překrytím osob dokáže úspěšně vyrovnat.

Ukázka správné funkce aplikace je na obr. 26. Všechny postavy jsou korektně označeny a nedochází k jejich záměně.



Obrázek 26: Správná funkce aplikace.

Testování funkce počítání proběhlo pouze na omezené sadě videozáznamů. Z výše uvedeného však přesto vyplývá, že přesnost počítání silně závisí na podmínkách snímané scény. Aplikace by našla uplatnění především ve vnitřních prostorech bez častých a výrazných změn pozadí. Umístění kamery a úhel záběru by bylo vhodné směrovat do prostoru, kde se lidé pohybují většinou jednotlivě. Ideálním místem by byl například prostor turniketu. V opačném případě, a bez dalšího vylepšení segmentace postav, nebude počítání dosahovat kvalitních výsledků.

5 Závěr

V úvodu práce byla zmíněna motivace pro vytváření inteligentních kamerových systémů. Dále byly představeny metody a postupy zpracování obrazu pokrývající oblast detekce a sledování osob (kap. 2). Tato část práce posloužila jako teoretický základ pro následnou praktickou část, během které byla vytvořena aplikace pro počítání osob ve vstupní videosekvenci.

Při samotném vývoji aplikace byly nejdříve otestovány metody segmentace postav. Byly vyzkoušeny deskriptory pro detekci postav Histogram of Oriented Gradients a Viola & Jones implementované v multiplatformní knihovně OpenCV. Po tomto přehledovém testování metod bylo upuštěno od původnímu záměru využití metody HOG. Důvodem byla především výpočetní náročnost metody. Segmentace postav je v aplikaci prováděna pouze pomocí detekce pohybu metodou Mixture of Gaussians z knihovny OpenCV. Pro sledování postav ve videosekvenci je využita predikční schopnost Kalmanova filtru a v případě částečného překrytí postav i metoda Template Matching. Tím došlo ke zvýšení robustnosti algoritmu právě v situacích, kdy se postavy míjejí a vzájemně překrývají. Na základě sledování pohybu osob je prováděno počítání osob, které překročí uživatelem stanovenou linii. Popis zvolených metod a výsledný algoritmus byl uveden v kap.3.

Pro testování aplikace byl použit dataset PETS 2006. Testování aplikace potvrdilo předpokládané slabé místo algoritmu, kterým je segmentace a následné sledování skupiny osob. Naopak výsledky pro osoby, které do záznamu vstoupily samostatně, byly kvalitní (kap. 4.4). Případná další práce by tedy měla směřovat zejména ke zlepšení segmentace jednotlivých osob.

6 Reference

[1] NetRex [online], *Webová prezentace společnosti NetRex*, 2015, [cit. 2015-10-2], Dostupné z WWW: http://www.netrex.cz/cz/sluzby-a-produkty/people-counting/metody/.

[2] Intya Counter [online], *Webová prezentace společnosti Ronyo Technologies*, 2013, [cit. 2015-10-2], Dostupné z WWW:http://ronyo.cz/reseni/person-counter-hw-pocitani-osob/.

[3] Ch. Stauffer and W. E. L. Grimson, *Adaptive background mixture models for real-time tracking*, In Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on., volume 2. IEEE, 1999.

[4] T. F. Cootes , C. J. Taylor, *Active Shape Models - Smart Snakes* ,in Proceedings of the British Machine Vision Conference, 1992.

[5] P. Viola, M. J. Jones, *Robust Real-time Object Detection*,Cambridge Research Laboratory, Cambridge , 2001.

[6] E. Sojka, *Digitální zpracování obrazu a analýza obrazu*, VŠB–TU ostrava, Fakulta elektrotechniky a informatiky, Ostrava, 2000.

[7] S. Russell,P. Norvig, *Artificial Intelligence, A Modern Approach*, Prentice Hall, 1995

[8] N. Dalal, *Finding people in images and videos*, Institut Nationale Polytechnique de Grenoble, 2006.

[9] Z.Zivkovic, *Improved adaptive Gausian mixture model for background subtraction*, International Conference Pattern Recognition, UK, 2004.

[10] A. Williams, D. Ganesan, and A. Hanson, *Aging inplace: fall detection and localization in a distributed smart camera network*, In MULTIMEDIA'07, 2007.

[11] David Nicholas Olivieri, Iván Gómez Conde, Xosé Antón Vila Sobrino, *Eigenspace-based fall detection and activity recognition from motion templates and machine learning*, Expert Systems with Applications, Volume 39, Issue 5, 2012.

A Ukázka vykreslení trajektorie

Obrázek 27: Ukázka vykreslení trajektorie.

B Obsah přiloženého CD

Čítání osob ve videosekvencích.pdf
Textová část diplomové práce.

readme.txt
Soubor obsahuje důležité informace pro spuštění a správnou funkčnost aplikace „Tracking.exe“.

Tracking.exe
Spustitelný program.

Tconfig.txt
Konfigurační soubor aplikace.

dataset/
Složka obsahuje testovací data.

src/
Složka obsahuje zdrojové soubory aplikace.

knihovny OpenCV
`opencv_core248.dll`
`opencv_highgui248.dll`
`opencv_imgproc248.dll`
`opencv_video248.dll`